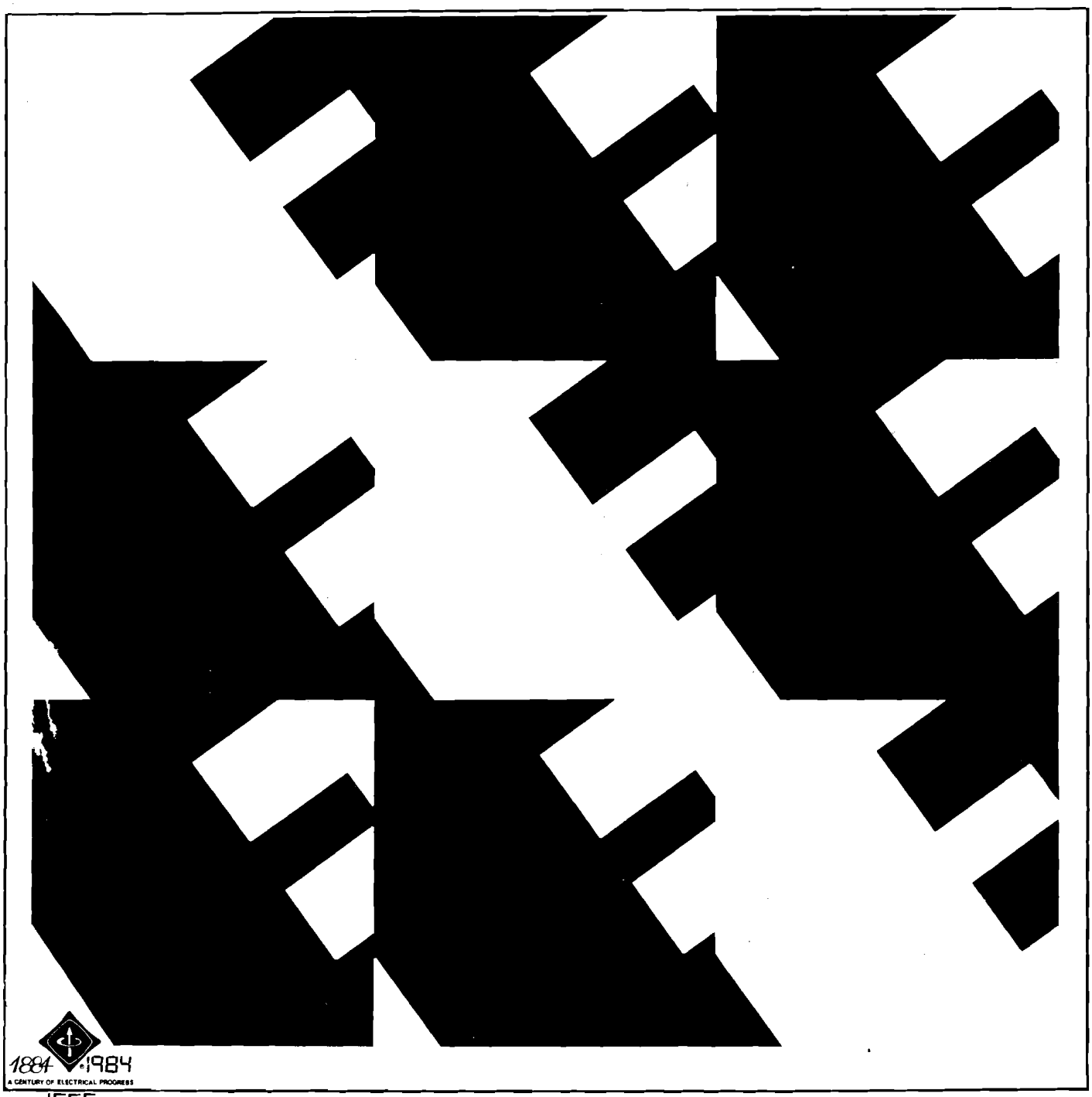
IEEE Standard
Microcomputer System Bus



Published by The Institute of Electrical and Electronics Engineers, Inc 345 East 47th Street, New York, NY 10017, USA

December 29, 1983

SH09001

IEEE Std 796-1983

IEEE
Std 796-1983

IEEE Standard
Microcomputer System Bus

Sponsor

Standards Committee of the
IEEE Computer Society

The Institute of Electrical and Electronics Engineers, Inc
345 East 47th Street, New York, NY 10017

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
345 East 47th Street
New York, NY 10017
USA

Foreword

(This Foreword is not a part of IEEE **Std 796-1983,** IEEE **Standard Microcomputer System Bus.)**

The IEEE Std 796 bus is a commercial-quality bus for use in a microcomputer-based system. The board modules that connect to the bus may be slaves (memory or input/output, or both), masters (CPU or controllers, or both), or both.

The IEEE Std 796 bus provides an electrical and mechanical specification to allow many different manufacturers to produce varied but compatible microcomputer modules. This offers the user the flexibility to choose the modules that solve his microcomputer system problem in the most complete and cost-effective manner.

The bus is a departure from the minicomputer market, where each major manufacturer defines its bus without regard for its potential use by other systems or manufacturers. The IEEE Std 796 bus, though initiated by Intel as their Multibus, is a cooperative industry effort toward establishing a standard for a large number of manufacturers and users of microcomputer modules.

The original development of the Multibus was done at Intel by their initial systems engineering group—Bob Garrow, Rich Boberg, Hap Walker, and Mike Yen—with significant input by Fred Coury, a consultant. The bus was designated the Multibus, an Intel trademark, and was used as the basis for Intel's Intellec Microcomputer Development System introduced in 1975. Intel decided to use the same bus for its subsequent iSBC product line. The success of these reliable OEM board products produced hundreds of competitive and complementary products, all made to be compatible with this commercialquality bus. This diverse offering of modules compatible at the board level has provided customers with a comprehensive and cost-effective product selection--even with second-sourcing at the board level. The customers of these boards are companies whose value-added consists of their own special-purpose hardware and software; they use the standard CPU, memory, I/O, and peripheral controller modules as the basis for their systems. The version of the bus standardized by the IEEE will help assure the user community that the compatibility and high commercial quality of the bus will be maintained from one manufacturer to another.

This edition of IEEE Std 796 is complete and accurate, however, as expected, the IEEE Std 796 bus will continue to evolve with enhancements and refinements. The general philosophy being followed for these changes is to maintain upward and downward compatibility, where possible, while creating a clean, professional standard. In some cases, the IEEE Std 796 Working Group has elected to specify the bus in a way that will cause technical incompatibility with existing products. The removal of the –5 volt power supply from the bus pins is an example. In these cases, it is the opinion of the committee that there is minimal impact since the products can still be used (by making exceptions to the standard) until the products are changed or their lifetimes expire. This temporary inconvenience permits the deliberate focusing of all new or redesigned products onto the preferred standard.

The IEEE Standards Board calls attention to the fact that it is claimed that there is a patent on the common bus request owned by Intel Corporation and that Intel has patents pending on the byte-swap aspects of the bus. These patents appear to cover the subject, as partially described in 2.1.3.2.2 and 2.2.2.4. IEEE takes no position with respect to patent validity. Intel Corporation has assured the IEEE that it is willing to grant a license on this aspect of the bus on reasonable and nondiscriminatory terms to anyone wishing to obtain such a license. Intel Corporation's undertakings in this respect are on file with the IEEE Standards Office, and the license details may be obtained from the legal department of Intel Corporation whose address is Intel Corporation, 5200 N.E. Elam Young Parkway, Hillsboro, Oregon 97123.

Although the original work in creating the IEEE Std 796 bus was done at Intel Corporation, the current specification represents the work of individuals from many different companies. These individuals include the members of the IEEE Std 796 Working Group of the IEEE Computer Society's Microprocessor Standards Committee.

At the time this standard was approved the membership of the working group was as follows.

Richard W. **Boberg,** *Chairman*

Rod H. Allen
Mark Bagula
Ron Dilbeck
Gary **Fieland**
Robert **Garrow**
Bill Halloway
Jim Johnson
Jim Kelley
Craig Kinnie
Jim Konsevich
Larry Michel
Martin A. **Newman**
Tung-sun Tung

When the IEEE Standards Board approved this standard on December 9,1982, it had the following membership:

I. N. Howell, Jr, *Chairman*

Edward Chelotti, *Vice Chairman*

Sava I. **Sherr,** *Secretary*

G. Y. R. Allen
J. J. Archambault
James H. **Beall**
John T. Boettger
J. V. Bonucchi
Edward J. Cohen
Len S. Corey
Donald C. Fleckenstein
Jay Forster
Kurt Greene
Joseph L. Koepfinger
Irving Kolodny
John E. May
Donald T. Michael*
A. R. Parsons
J. P. Riganati
Frank L. Rose
Robert W. Seelbach
Jay A. Stewart
Clifford O. **Swanson**
Robert E. Weiler

***Member emeritus**

Contents

SECTION PAGE

1. General 9
1.1 Scope 9
1.2 Object 9
1.3 Definitions 10
1.3.1 General System Terms 10
1.3.2 Signals and Paths 10

2. Functional Description 10
2.1 IEEE Std 796 Bus Elements 11
2.1.1 Masters 11
2.1.2 Slaves 11
2.1.3 IEEE Std 796 Bus Signals 12
2.1.3.1 Control Lines 12
2.1.3.1.1 Clock Lines 12
2.1.3.1.2 Command Lines (MWTC*, MRDC*, IOWC*, IORC*) 12
2.1.3.1.3 Transfer Acknowledge Line (XACK*) 12
2.1.3.1.4 Initialize (INIT*) 12
2.1.3.1.5 Lock (LOCK*) 12
2.1.3.2 Address and Inhibit Lines 12
2.1.3.2.1 Address Lines (A0 - A23*) 13
2.1.3.2.2 Byte High Enable Line (BHEN*) 13
2.1.3.2.3 Inhibit Lines (INH1* and INH2*) 13
2.1.3.3 Data Lines (D0* - D15*) 13
2.1.3.4 Interrupt Lines 13
2.1.3.4.1 Interrupt Request Lines (INTO* - INT7*) 13
2.1.3.4.2 Interrupt Acknowledge (INTA*) 13
2.1.3.5 Bus Exchange Lines 13
2.1.3.5.1 Bus Request (BREQ*) 13
2.1.3.5.2 Bus Priority (BPRN* and BPRO*) 13
2.1.3.5.3 Bus Busy (BUSY*) 13
2.1.3.5.4 Common Bus Request (CBRQ*) 13
2.2 Data Transfer Operation 14
2.2.1 Data Transfer Overview 14
2.2.2 Signal Descriptions 15
2.2.2.1 Initialize (INIT*) 15
2.2.2.2 Constant Clock (CCLK*) 15
2.2.2.3 Address Lines (A0* - A23*) 15
2.2.2.4 Data Lines (DO* - D15*) 16
2.2.2.5 IEEE Std 796 Bus Commands 18
2.2.2.5.1 Read Operation 18
2.2.2.5.2 Write Operation 18
2.2.2.6 Transfer Acknowledge (XACK*) 19
2.2.2.7 Inhibit (INH1 and INH2*) 19
2.2.2.8 Lock (LOCK*) 21
2.3 Interrupt Operations 22
2.3.1 Interrupt Signal Lines 22
2.3.1.1 Interrupt Request Lines (INTO* - INT7*) 22
2.3.1.2 Interrupt Acknowledge (INTA*) 22
2.3.2 Classes of Interrupt Implementation 22
2.3.2.1 Non-Bus Vectored Interrupts 22
2.3.2.2 Bus Vectored Interrupts 23

SECTION PAGE

2.4 IEEE Std 796 Bus Exchange 24
2.4.1 IEEE Std 796 Bus Exchange Signals 25
2.4.1.1 Bus Clock (BCLK*) 25
2.4.1.2 Bus Busy (BUSY*) 25
2.4.1.3 Bus Priority In (BPRN*) 25
2.4.1.4 Bus Priority Out (BPRO*) 25
2.4.1.5 Bus Request (BREQ*) 25
2.4.1.6 Common Bus Request (CBRQ*) 25
2.4.2 Bus Exchange Priority Techniques 25
2.4.2.1 Serial Priority Technique 26
2.4.2.2 Parallel Arbitration Technique 27

3. Electrical Specifications 27
3.1 General Bus Considerations 27
3.1.1 Logical and Electrical State Relationships 27
3.1.2 Signal Line Characteristics 27
3.1.2.1 In-Use Signal Line Requirements 28
3.1.2.2 Backplane Signal Trace Characteristics 29
3.1.3 Power Supply Specifications 29
3.1.4 Temperature and Humidity 29
3.2 Timing 31
3.2.1 Read Operations (I/O and Memory) 31
3.2.2 Write Operations (I/O and Memory) 34
3.2.3 Inhibit Operations 34
3.2.4 Interrupt Implementations 34
3.2.4.1 NBV Interrupts 34
3.2.4.2 BV Interrupts 34
3.2.5 Bus Control Exchanges 36
3.2.5.1 Serial Priority 36
3.2.5.2 Parallel Priority 37
3.2.6 Miscellaneous Timing 37
3.3 Receiver Modules, Driver Modules, and Terminations 37

4. Mechanical Specifications 40
4.1 Backplane Considerations 40
4.1.1 Board to Board Relationships 40
4.1.2 IEEE Std 796 Bus Pin Assignments 43
4.2 IEEE Std 796 Bus Board Form Factors 43
4.2.1 Connector Naming and Pin Numbering Standards 43
4.2.2 Standard Outline of Printed Wiring Boards 43
4.2.3 Bus Connectors 43

5. Levels of Compliance 44
5.1 Variable Elements of Capability 44
5.1.1 Data Path 45
5.1.2 Memory Address Path 45
5.1.3 I/O Address Path 45
5.1.4 Interrupt Attributes 45
5.2 Masters and Slaves 45
5.3 Compliance Level Notation 45
5.3.1 Data Path 45
5.3.2 Memory Address Path 45
5.3.3 I/O Address Path 45
5.3.4 Interrupt Attributes 45
5.3.5 An Example 46
5.3.6 Compliance Marking 46

SECTION PAGE

6. Bibliography 46

FIGURES

Fig 1 IEEE Std 796 Bus Master and Slave Example 11
Fig 2 IEEE Std 796 Bus Interface Lines 14
Fig 3 IEEE Std 796 Bus Read Operation 15
Fig 4 IEEE Std 796 Bus Write Operation 15
Fig 5 IEEE Std 796 Bus Address Line Usage 16
Fig 6 IEEE Std 796 Bus Data Line Usage 17
Fig 7 Memory or I/O Read Timing 18
Fig 8 Memory or I/O Write Timing 19
Fig 9 Inhibit Timing for Write Operation 20
Fig 10 IEEE Std 796 Bus Lock Usage 21
Fig 11 Lock Timing 22
Fig 12 Non-Bus Vectored (BV) Interrupt Logic 23
Fig 13 Bus Vectored (BV) Interrupt Logic 24
Fig 14 Serial Priority Technique 26
Fig 15 Parallel Priority Technique 26
Fig 16 Setup. Hold. and Command Ringing Summary 28
Fig 17 Line-to-Line Coupling Characteristics 29
Fig 18 Read AC Timing 31
Fig 19 Write AC Timing 33
Fig 20 Inhibit AC Timing 33
Fig 21 Bus Vectored (BV) Interrupt AC Timing 34
Fig 22 Bus Exchange AC Timing 35
Fig 23 Common Bus Request AC Timing 36
Fig 24 Serial Priority AC Timing 37
Fig 25 Parallel Priority AC Timing 38
Fig 26 Constant Clock AC Timing 38
Fig 27 Command Separation AC Timing 39
Fig 28 Initialize AC Timing 39
Fig 29 Lock AC Timing 39
Fig 30 IEEE Std 796 Bus Backplane Card to Cage Separation 40
Fig 31 Typical IEEE Std 796 Bus Backplane 42
Fig 32 Connector and Pin Numbering 43
Fig 33 Standard Printed Wiring Board Outline 44

TABLES

Table 1 Electrical and Logic State Definition 10
Table 2 Control Line Signals 12
Table 3 Logical/Electrical State Relationships for "*" Signal 27
Table 4 Logical/Electrical State Relationships for Non "*" Signal 27
Table 5 Typical Rise and Fall Times 28
Table 6 IEEE Std 796 Bus Power Supply Specifications 29
Table 7 IEEE Std 796 Bus Timing Specifications Summary 30
Table 8 Bus Drivers. Receivers. and Terminations 32
Table 9 Pin Assignments of Bus Signals on IEEE Std 796 Bus Board Connector (P1) 41
Table 10 Pin Assignments of Bus Signals on IEEE Std 796 Bus Board Connector (P2) 42

IEEE Standard
Microcomputer System Bus

1. General

1.1 Scope. One of the most important elements in a computer system is the bus structure that supplies the interface for all the hardware components. This bus structure contains the necessary signals to allow the various system components to interact with each other. It allows memory and I/O data transfers, direct memory accesses, generation of interrupts, etc. This standard provides a detailed description of all the elements and features that make up the IEEE Std 796 bus.

The bus supports two independent address spaces: memory and I/O. During memory cycles, the bus allows direct addressability of up to 16 megabytes using 24-bit addressing. During I/O bus cycles, the bus allows addressing of up to 64 kilobyte I/O ports using 16-bit addressing. Memory and I/O cycles can support 8-bit or 16-bit data transfers.

The bus structure is built upon the master-slave concept where the master device in the system takes control of the bus and the slave device, upon decoding its address, and acts upon the command provided by the master. This handshake (master-slave relationship) between the master and slave devices allows modules of different speeds to be interfaced by way of the bus. It also allows data rates up to five million transfers per second (bytes or words) to take place across the bus.

Another important feature of the bus is the ability to connect multiple master modules for multiprocessing configurations. The bus provides control signals for connecting multiple masters in either a serial or parallel priority fashion. With either of these two arrangements, more than one master may share bus resources.

This standard has been prepared for those users who evaluate or design products that will be compatible with the IEEE Std 796 system bus structure. To this end, the necessary signal definitions and timing and electrical specifications have been covered in detail.

This standard deals only with the interface characteristics of microcomputer devices, not with design specifications, performance requirements, and safety requirements of modules.

Throughout this standard, the term ***system*** denotes the byte or word interface system that, in general, includes all the circuits, connectors, and control protocol to effect unambiguous data transfer between devices. The term ***device*** or ***module*** denotes any product connected to the interface system that communicates information by way of the bus, and that conforms to the interface system definition.

1.2 Object. The objective of this standard is to

(1) Define a general purpose microcomputer system bus.

(2) Specify the device-independent electrical and functional interface requirements that a module shall meet in order to interconnect and communicate unambiguously by way of the system.

(3) Specify the terminology and definitions related to the system.

(4) Enable the interconnection of independently manufactured devices into a single functional system.

(5) Permit products with a wide range of capabilities to be introduced to the system simultaneously.

(6) Define a system with a minimum of restrictions on the performance characteristics of devices connected to the system.

1.3 Definitions. The following general definitions apply throughout this standard. More detailed definitions can be found in the appropriate section.

1.3.1 General System Terms

compatibility. The degree to which devices may be interconnected and used without modification, when designed as defined in Sections 2 and **3** of this standard. Section 5 introduces the notion of levels of compliance and the corresponding notation.

bus cycle. The process whereby digital signals effect the transfer of data bytes or words across the interface by means of an interlocked sequence of control signals. ***Interlocked*** denotes a fixed sequence of events in which one event must occur before the next event can occur.

interface. **A** shared boundary between two systems, or between parts of systems, through which information is conveyed.

interface system. The device-dependent electrical and functional interface elements necessary for communication between devices. Typical elements are: driver and receiver circuits, signal line descriptions, timing and control conventions, and functional logic circuits.

override. **A** bus master overrides the bus control logic when it is necessary to guarantee itself back-to-back bus cycles. This is called ***overriding*** the bus, temporarily preventing other masters from using the bus.

system. **A** set of interconnected elements which achieve a given objective through the performance of a specified function.

1.3.2 Signals and Paths

bus. **A** signal line or a set of lines used by an interface system to connect a number of devices and to transfer information.

byte. **A** group of eight adjacent bits operated on as a unit.

word. Two bytes or sixteen bits operated on as a unit.

high state. The more positive voltage level used to represent one of two logical binary states.

low state. The more negative voltage level used to represent one of two logical binary states.

signal. The physical representation of data.

signal level. The relative magnitude of a signal when compared to an arbitrary reference. Signal levels in this standard are specified in volts.

signal line. One of a set of signal conductors in an interface system used to transfer messages among interconnected devices.

signal parameter. That element of an electrical quantity whose values or sequence of values convey information.

2. Functional Description

This section provides an overall understanding of how the IEEE Std **796** bus functions and describes the elements that connect to the bus, the signals that provide the interface to the bus, and the different types of operations performed on the bus.

In this section, as well as throughout the standard, a clear and consistent notation for signals has been used. The memory write command (MWTC) is used to explain this notation. The terms one:zero and true:false can be ambiguous, so their use is avoided. In their place, the terms electrical high and low (H and L) are used. A nathan (asterisk) following the signal name, MWTC*, indicates that the signal is active low as shown:

MWTC* = asserted at 0 volts

The signal (MWTC*) driven by a three-state driver will be pulled up to V_{CC} when not asserted. To further explain the notation used in this standard see Table **1.**

Table **1**
Electrical **and** Logic State Definition

Function	Definition Electrical	Logic	State
MWTC	**H**	**1 True**	**Active, asserted**
	L	**0 False**	
MWTC*	**L**	**1 True**	**Active, asserted**
	H	**0 False**	

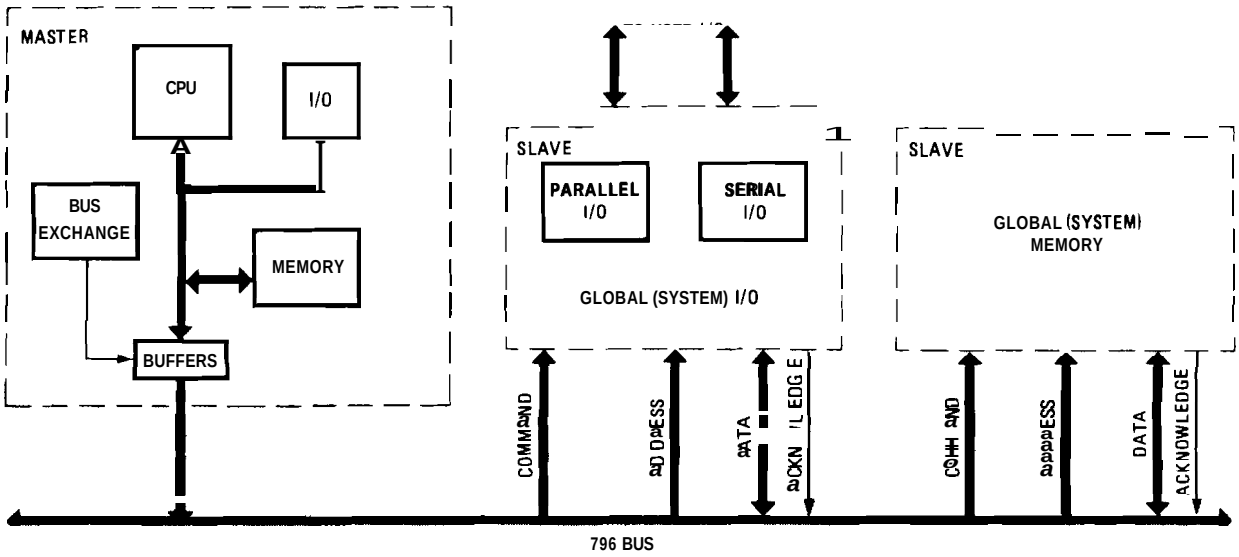


Fig 1
IEEE Std 796 Bus Master and Slave Example

2.1 IEEE Std 796 Bus Elements. This subsection describes the elements, masters and slaves, that interface to the bus and the IEEE Std 796 bus signal lines that comprise this interface.

2.1.1 Masters. A master is any module having the ability to control the bus. The master exercises this control by acquiring the bus through bus exchange logic and then generating command signals, address signals, and memory or I/O addresses. To perform these tasks, the master is equipped with either a central processing unit or logic dedicated to transferring data over to the bus and to and from other destinations. Figure 1 depicts a system that includes a master and two slave models.

The IEEE Std 796 bus architecture can support more than one master in the same system, but to do this, there should be a means for each master to gain control of the bus. This is accomplished through the bus exchange logic (see 2.4).

Masters may operate in one of two modes of operation. Modes 1 and 2 are defined as follows:

Mode 1: Masters are limited to single bus transfers per bus connect. If all masters are Mode 1, system timing is rendered deterministic by conformance with a maximum bus busy period. That period is limited by the parameter t_{BYSO} maximum (see 3.2.5).

Mode 2: Masters are unlimited in this bus control. They may invoke bus override. Bus timeouts are allowed. Conformance with the maximum busy period is not required. The last classification is included to allow for a very broad class of operations, giving users maximum flexibility in meeting these applications' needs. The first mode of operation is defined to allow system designers to predict the overall performance of their systems without concern for uncontrolled timing parameters such as bus timeout. For masters which can only operate in Mode 2, their specification shall state "Mode 2 master only."

2.1.2 Slaves. Another type of module that can interface to the bus is the slave. Slave modules decode the address lines and act upon the command signals from the masters. The slaves are not capable of controlling the bus. Some examples of bus slaves are shown in Fig 1.

Table 2
Control Line Signals

Class	Function	Signal
Clocks	Constant clock	CCLK*
	Bus clock	BCLK*
Commands	Memory write	MWTC*
	Memory read	MRDC*
	I/O write	IOWC*
	I/O read	IORC*
Acknowledge	Transfer acknowledge	XACK*
Initialize		INIT*
Lock		LOCK*

2.1.3 IEEE Std 796 Bus Signals. Signals transferred over the bus can be grouped into several classes based on the functions they perform. The classes are as follows:

(1) Control lines
(2) Address and inhibit lines
(3) Data lines
(4) Interrupt lines
(5) Bus exchange lines

The following subsections explain the different classes of IEEE Std 796 bus signals.

2.1.3.1 Control Lines. Signals classified as control lines are listed in Table 2.

2.1.3.1.1 Clock Lines.

(1) Bus Clock (BCLK*). A periodic signal used to synchronize the bus contention logic; it may be slowed, stopped, or single stepped. The bus clock shall be generated by one and only one source within the system. This means that each standalone hus master must have the capability of generating an acceptable clock that can optionally be connected to, or disconnected from, the bus. In a multimaster system, only one of the masters shall have its clock connected to the bus.

(2) Constant Clock (CCLK*). A periodic signal of constant frequency, which may be used by masters or slaves as a master clock. The constant clock shall be generated by one and only one source within the system. This means that each bus master should have the capability of generating an acceptable clock that can optionally be connected to, or disconnected from, the bus. In a multimaster system, only one of the masters shall have its clock connected to the bus.

2.1.3.1.2 Command Lines (MWTC*, MRDC*, IOWC*, IORC*). The command lines are elements of a communication link between the masters and slaves. There are two command lines for memory and two command lines for I/O. An active command line indicates to the slave that the address lines are carrying a valid address and that the slave is to perform the specified operation. In a data-write cycle, the active command line (MWTC* or IOWC*) additionally indicates that the data is valid on the bus. In a data-read cycle, the transition of the command (MRDC* or IORC*) from active to inactive indicates that the master has received the data from the slave.

2.1.3.1.3 Transfer Acknowledge Line (XACK*). This line is used by the slaves to acknowledge commands from the master. XACK* indicates to the master that the requested action is complete and that data has been placed on, or accepted from, the data lines.

2.1.3.1.4 Initialize (INIT*). The INIT* signal is generated to reset the entire system to a known internal state. This signal is usually generated prior to starting any operations on the system. INIT* may be generated by any or all of the bus masters or by an external source such as a buffered and debounced front panel switch.

2.1.3.1.5 Lock (LOCK*). The LOCK signal is generated by the master in control of the bus to indicate the slave is locked. LOCK* is used to extend mutual exclusion to multiple port RAM designs.

2.1.3.2 Address and Inhibit Lines. The address and inhibit lines are used for the following signals:

Function	Signal
Address lines	A0*–A23*
Byte high enable	BHEN*
Inhibit lines	INH1* and INH2*

2.1.3.2.1 Address Lines (A0–A23*). These lines, which specify the address of the referenced memory location or I/O device, allow a maximum of 16 megabytes (16 777 216 bytes) of memory to be accessed. When addressing an I/O device, a maximum of 16 address lines (A0*–A15*) is used, thus allowing the addressing of a maximum of 64 kilobyte devices. An I/O module should also be able to be configured to decode only eight address lines (A0*–A7*) and ignore the upper eight lines (see 2.2.2.3).

2.1.3.2.2 Byte High Enable Line (BHEN*). This byte control line is used to enable the upper byte (bits 8–15) of a 16-bit word to drive the bus. The signal is used only on systems that incorporate 16-bit data transfers.

2.1.3.2.3 Inhibit Lines (INH1* and INH2*). The inhibit lines can be invoked for any memory read or memory write operation (MRDC* or MWTC*). An inhibit line is asserted by a slave to inhibit another slave's bus activity during a memory read or write operation. The inhibit signal generated by the inhibiting slave is derived from decoding the memory address lines. The inhibiting slave can decode a single address, a block of addresses, or any combination of single and block addresses.

When it detects the specific address during an actual command (MRDC* or MWTC*), the inhibiting slave generates an inhibit signal, which is sensed by the inhibited slave. When so inhibited, this slave module disables its drivers from all data, address, and acknowledge bus lines, although it may actually perform internal operations. [All modules that may be inhibited should have completed internal operations within 1.5 μs from the start of the command line. This interval (1.5 μs) is also the minimum acknowledge timing for modules issuing inhibits. This guarantees that inhibited modules have sufficient time to return to their normal state before the current bus command is completed.]

2.1.3.3 Data Lines (D0*–D15*). These 16 bidirectional data lines transmit and receive information to and from a memory location or an I/O port. (D15* is the most significant bit and DO* is the least significant bit). In 8-bit transfer, only lines DO*–D7* are valid.

2.1.3.4 Interrupt Lines. The' interrupt lines consist of the following signals:

Function	Signal
Interrupt requests	INTO*–INT7*
Interrupt acknowledge	INTA*

2.1.3.4.1 Interrupt Request Lines (INTO*–INT7*). Interrupts are requested by activating one of the eight interrupt request lines. INTO* has the highest priority and INT7* has the lowest priority.

2.1.3.4.2 Interrupt Acknowledge (INTA*). In response to an interrupt request signal, an interrupt acknowledge signal can be generated by a bus master with bus vectored interrupt capability. The interrupt acknowledge signal is used to freeze the interrupt status and request the placement of the interrupt vector address on the bus data lines.

2.1.3.5 Bus Exchange Lines. The bus exchange lines are used by the following signals:

Function	Signal
Bus clock	BCLK*
Bus request	BREQ*
Bus priority	BPRN*, BPRO*
Bus busy	BUSY*
Common bus request	CBRQ*

A master gains control of the bus through the manipulation of these signals.

2.1.3.5.1 Bus Request (BREQ*). A signal used by the bus masters in a priority resolution circuit to indicate a request for control of the bus.

2.1.3.5.2 Bus Priority (BPRN* and BPRO*). The priority functions allow masters to break deadlocks that occur when more than one master concurrently requests the bus. The bus priority in (BPRN*) signal indicates to a particular master that no higher priority master is requesting use of the bus. The bus priority out (BPRO*) signal is used in serial (daisy chain) bus priority resolution schemes. In such a scheme, BPRO* is passed by one master to the BPRN* input of the master with the next lower bus priority; when active, the BPRO* signal indicates that the higher priority master does not require control of the bus.

2.1.3.5.3 Bus Busy (BUSY*). A signal activated by the master in control of the bus to indicate that the bus is in use. This prevents other masters from gaining control of the bus.

2.1.3.5.4 Common Bus Request (CBRQ*). A signal that maximizes a master's data transfer

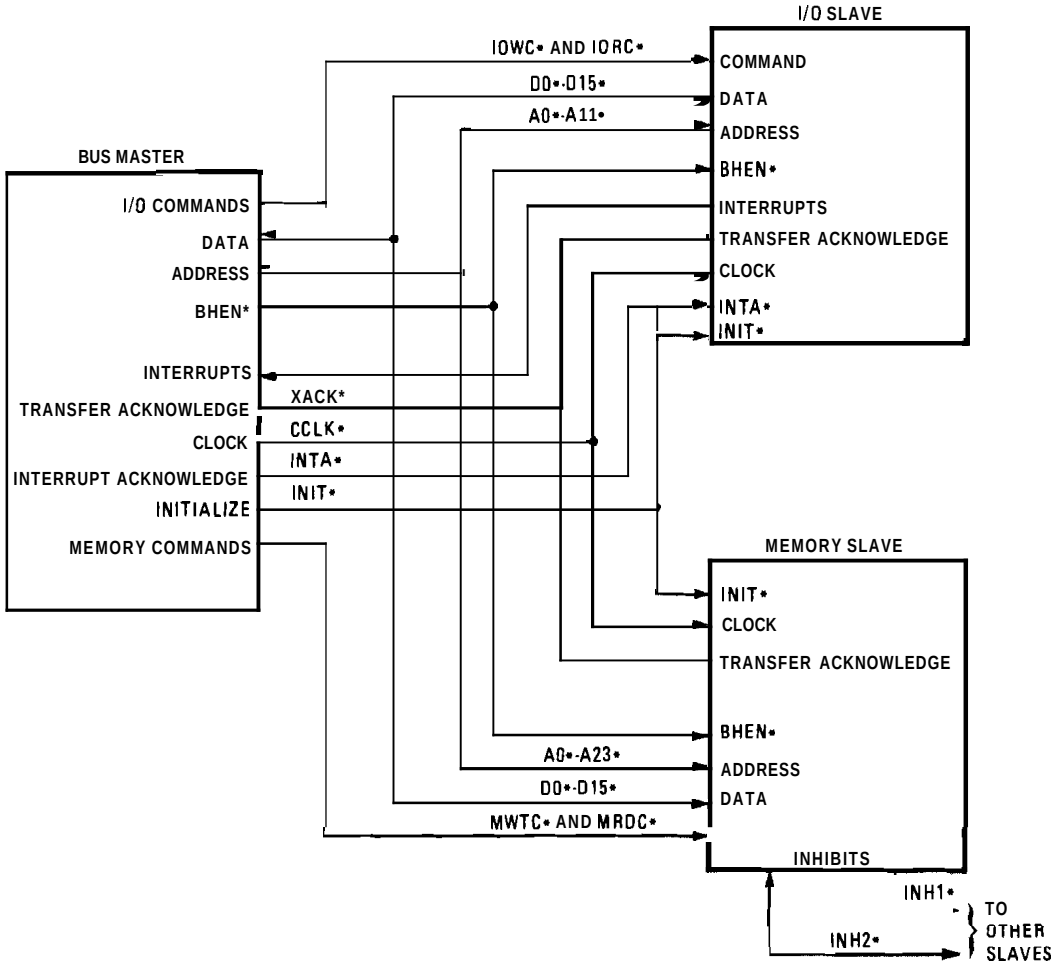


Fig 2
IEEE Std 796 Bus Interface Lines

rate to the bus by sensing the absence of other bus requests. The CBRQ* signal does this by serving two functions. It indicates to the master controlling the bus whether or not another master needs to gain control of the bus. For the other masters, it is a means of notifying the controlling bus master that it must relinquish control of the bus if it is not using the bus.

2.2 Data Transfer Operation. The primary function of the IEEE Std 796 bus architecture is to provide a path for the transfer of data between modules on the bus. The following subsections describe the different types of data transfers and the means by which they are implemented using the signals previously described. Figure 2 can be referenced during the following discussion.

The discussion of the data transfer operation of the bus is covered in three parts

(1) An overview of the operation

(2) A detailed description of the signals used in the transfer

(3) A discussion of the specifics pertaining to the different transfers

It is assumed in this discussion that there is only one master on the bus, and therefore no bus contention exists. The bus exchange logic is discussed in 2.4.

2.2.1 Data Transfer Overview. A data transfer is accomplished as follows: the bus master places the memory address or I/O port address on the address lines. If the operation is a write, the data is also placed on the data lines at this time. The bus master then generates a command (I/O read or write, or memory read or write),

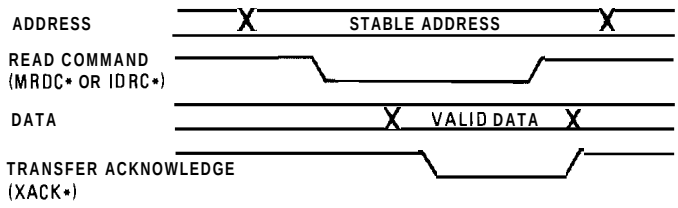


Fig 3
IEEE Std 796 Bus Read Operation

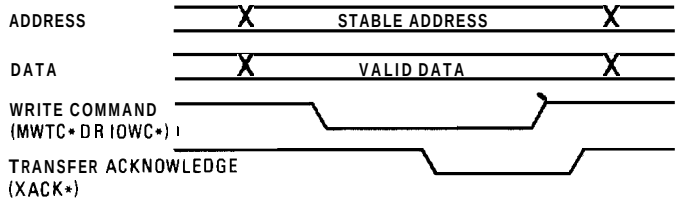


Fig 4
IEEE Std 796 Bus Write Operation

which activates the appropriate bus slave. The slave accepts the data if it is a write operation, or places the data on the data lines if it is a read operation. A transfer acknowledge signal is then sent to the bus master by the bus slave, allowing the bus master to complete its cycle by removing the command from the command line and then clearing the address and data lines. Figures 3 and 4 show the basic timing for a read and write data transfer operation.

2.2.2 Signal Descriptions. This subsection provides a detailed description of the IEEE Std 796 bus signals. Included are timing, signal origination, and other information pertaining to the specific function that each signal performs in the data transfer operation.

2.2.2.1 Initialize (INIT*). Prior to any operation of the bus, all system modules should be reset to a known internal state. This can be accomplished by an INIT* signal initiated by one of the following three sources:

(1) A power-on clear circuit (RC network), which holds INIT* low until the power supplies reach their specific voltage outputs.

(2) A reset button, which is sometimes provided on the system front panel for operator use. Note that this button should be debounced

(3) A software command that can be implemented to pull down the INIT* line.

The INIT* line is driven by open-collector gates and requires signal conditioning to meet the electrical specifications of the bus.

2.2.2.2 Constant Clock (CCLK*). The constant clock signal, which is driven by only one source, provides a timing source for any or all modules on the bus. CCLK* is a periodic signal with a specified frequency and is driven by a clock driver circuit.

2.2.2.3 Address Lines (A0*–A23*). The address lines are used to specify the address of the memory location or the I/O device that is being referenced by the command. There are 24 address lines, binary coded, to allow up to 16 777 216 bytes of memory to be referenced. These lines are driven by three-state drivers and are always controlled by the master using the bus.

For I/O bus cycles, master modules have the option of gathering 8-bit or 16-bit addresses. Because of this, all I/O slaves should be capable of being configured to decode eight address bits (A0*–A7*) and ignore the upper address bits or to decode all 16 bits of address (A0*–A15*). Note that in a system using 8-bit I/O

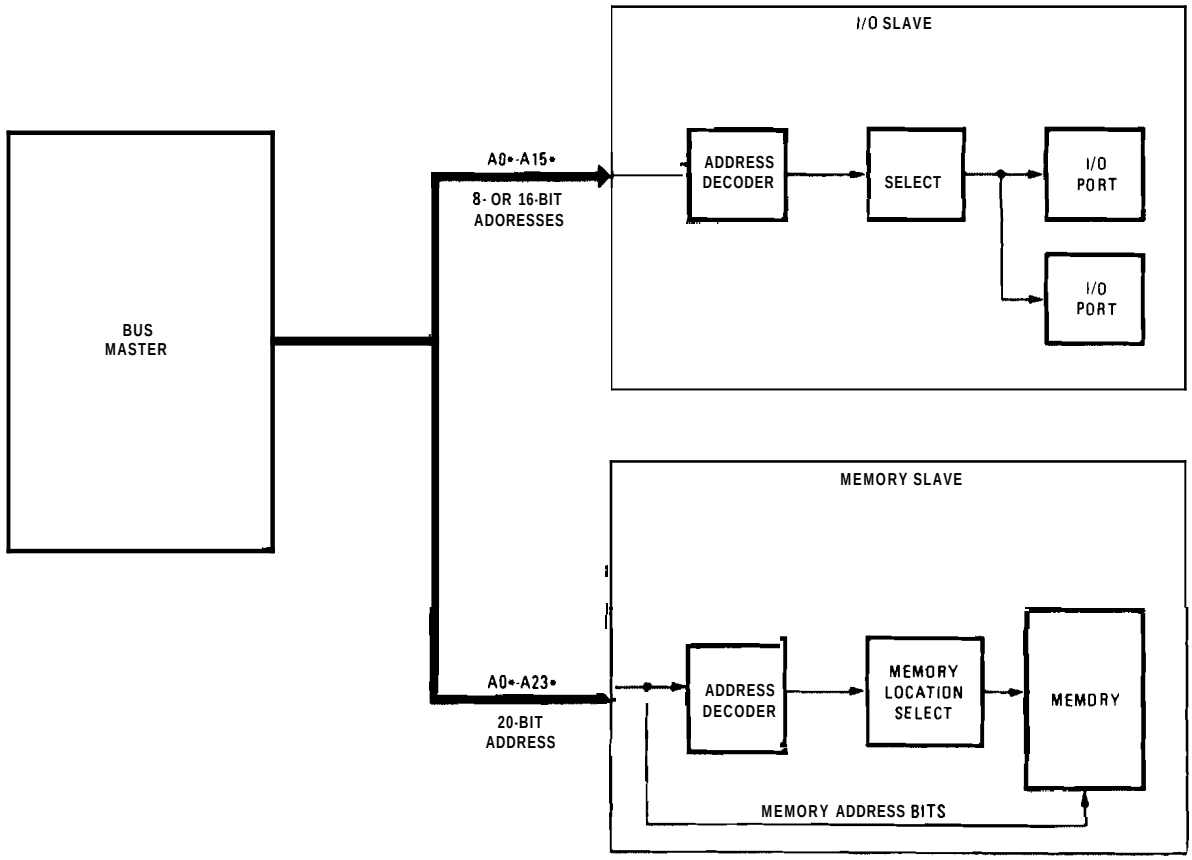


Fig 5
IEEE Std 796 Bus Address Line Usage

addresses, the value of the upper 16 bits of address is unknown. A master generating only 8-bit address may set the upper 16 address bits to any arbitrary value.

Refer to Fig 5 for an example of address line usage.

2.2.2.4 Data Lines (D0*–D15*). These are 16 bidirectional data lines used to transmit and receive information to and from a memory location or I/O port. The 16 lines are driven by the master on write operations and by the addressed slave, memory or I/O, on read operations. Both 16-bit and 8-bit transfers can be accomplished by using only lines D0*–D7*, with DO* being the least significant bit.

There are three types of transfers that take place across the bus

(1) Transfer of even byte on D0*–D7*

(2) Transfer of odd byte on D0*–D7*, using swap byte function

(3) Transfer of a 16-bit word

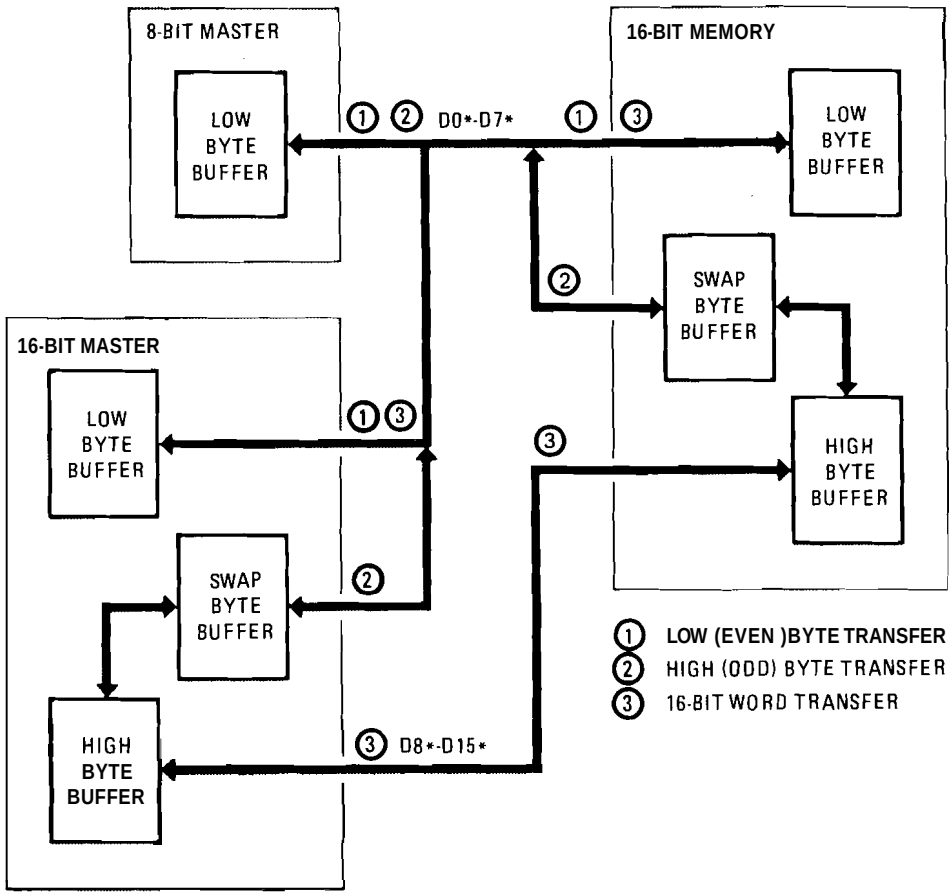


Fig 6
IEEE Std 796 Bus Data Line Usage

Figure 6 shows the data lines and the contents of these lines for the three types of transfers mentioned.

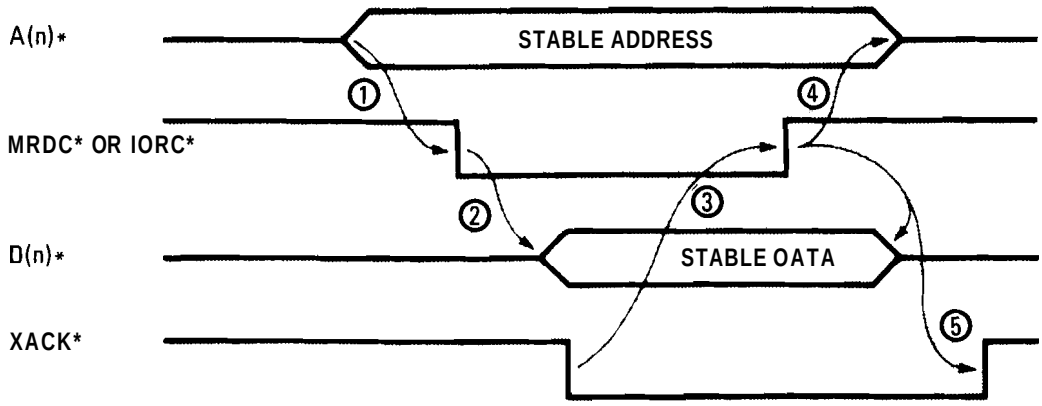
Two signals control the data transfers. Byte high enable (BHEN*) active indicates that the bus is operating in the 16-bit mode, and the address bit 0 (A0*) defines an even-type or odd-byte transfer.

For an even byte transfer, BHEN* and A0* are inactive, indicating the transfer of an even byte. The transfer takes place across data lines D0*–D7*.

For an odd-byte transfer, BHEN* is inactive and A0* is active, indicating the transfer of an odd byte. On this type of transfer, the odd byte is transferred through the swap byte buffer to D0*–D7*. The odd byte is transferred across on D0*–D7* to make 8-bit and 16-bit systems compatible.

For a 16-bit transfer, BHEN* is active and A0* is inactive. On this type of transfer, the even byte is transferred on D0*–D7* and the odd byte is transferred across the bus on D8*–D15*.

The IEEE Std 796 bus data lines are always driven by three-state drivers.



1 Address setup time: 50 ns minimum.
2 Time required for slave to get data onto bus in accordance with setup time requirement. XACK* can be asserted as soon as data is on bus.
3 Time required for master to remove command.
4 Address and data hold time: 50 ns minimum.
5 XACK* and data must be removed from the bus a maximum of 65 ns after the command is removed.

Fig 7
Memory or I/O Read Timing

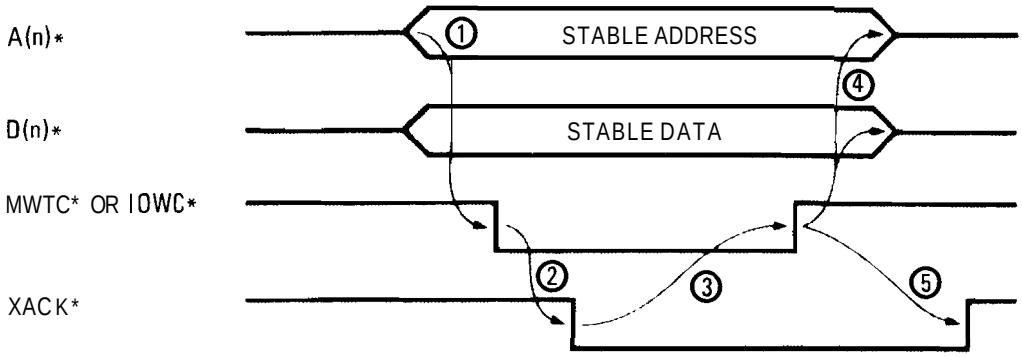
2.2.2.5 IEEE Std 796 Bus Commands. The command lines and how they work in conjunction with other lines to accomplish a read or a write operation are described in this subsection. There are four command lines

Function	Line
Memory read command	MRDC*
I/O read command	IORC*
Memory write command	MWTC*
I/O write command	IOWC*

The command lines, which are driven by three-state drivers on the bus master, indicate to the slave the action that is being requested.

2.2.2.5.1 Read Operation. The two read commands, MRDC* and IORC* initiate the same basic type of operation. The only difference is that MRDC* indicates that the memory address is valid on the address lines, whereas IORC* indicates that the I/O port address is valid on the address lines. This address, memory or I/O port, should be valid on the bus 50 ns prior to the read command being generated. When the read command is generated, the slave module, memory or I/O port, places the data on the data lines and returns a transfer acknowledge (XACK*) signal, indicating that the data is on the bus. When the bus master receives the acknowledge, it strobes in the data and removes the command, MRDC* or IORC*, from the bus. The slave address, memory or I/O port, remains valid on the bus a minimum of 50 ns after the read command is removed. XACK* should be removed from the bus within 65 ns after the command is removed to allow for the next bus cycle. Figure 7 shows the timing for the memory read or I/O read command.

2.2.2.5.2 Write Operation. The write commands, MWTC* and IOWC*, initiate the same basic type of operation. MWTC* indicates that the memory address is valid on the address lines, whereas IOWC* indicates that the I/O port address is valid on the address lines. The address, memory or I/O, and data should be valid on the bus 50 ns prior to the write command being generated. This requirement allows data to be latched on either the leading or trailing edge of the command. Whep the write command, MWTC* or IOWC*, is asserted, the data on the data lines is stable and can be accepted



1. **Address and data setup: 50 ns minimum.**
2. **Time required for slave to accept data.**
3. **Time required for master to remove command from bus.**
4. **Address and data hold time: 50 ns minimum.**
5. **XACK* must be off the bus 65 ns after command.**

Fig 8
Memory or I/O Write Timing

by the slave. The slave indicates acceptance of the data by returning a transfer acknowledge (XACK*), allowing the bus master to remove the command, address, and data from the bus. XACK* should be removed from the bus within 65 ns to allow for the next bus cycle. Figure 8 shows the timing for the memory write or I/O write command.

2.2.2.6 Transfer Acknowledge (XACK*). The transfer acknowledge (XACK*) signal is the response from the bus slave, memory or I/O, indicating that the commanded read or write operation is complete and that the data has been placed on, or accepted from, the data lines. In effect, this signal XACK* allows the bus master to complete the current bus cycle.

If a bus master addresses a nonexistent or malfunctioning memory or I/O module, an acknowledge will not be returned to the master. If this should occur, the bus master will normally wait indefinitely for an acknowledge and will therefore never relinquish control of the IEEE Std 796 bus. To avoid this possibility, a bus timeout function can optionally be implemented on a bus master to terminate a bus cycle after a preset interval, even if no acknowledge has been received. A bus timeout can therefore be defined as any data transfer cycle terminated by the master before the transfer acknowledge (XACK*) signal is received. The minimum allowable bus timeout interval is 1.0 ms.

2.2.2.7 Inhibit (INH1* and INH2*). The inhibit lines can be invoked for any memory-read or memory-write operation (MRDC* or MWTC*). An inhibit line is asserted by a slave to inhibit another slave's bus activity during a memory read or write operation. The inhibit signal generated by the inhibiting slave is derived from decoding the memory address lines (t_{ID} = 100 ns maximum). The inhibiting slave can decode a single address, a block of addresses, or any combination of single and block addresses.

When it detects the specific address during the actual command, MRDC* or MWTC*, the inhibiting slave generates an inhibit signal, which is sensed by the inhibited slave. When so inhibited, this slave module disables its drivers from all data, address, and acknowledge bus lines, although it may actually perform internal operations. [All modules that may be inhibited should have completed internal operations with 1.5 μs from the start of the command line. This interval (1.5 ps) is also the minimum acknowledge (t_{ACC}) timing for modules issuing inhibits. This guarantees inhibited modules sufficient time to return to their normal state before the current bus command is completed.]

The slaves involved in the inhibit operation fall into three inhibit classes: top (inhibit) priority, middle priority, and bottom priority. In reference to the above paragraphs, a higher priority slave module is the inhibiting slave and a lower priority slave is the inhibited slave. INH1*

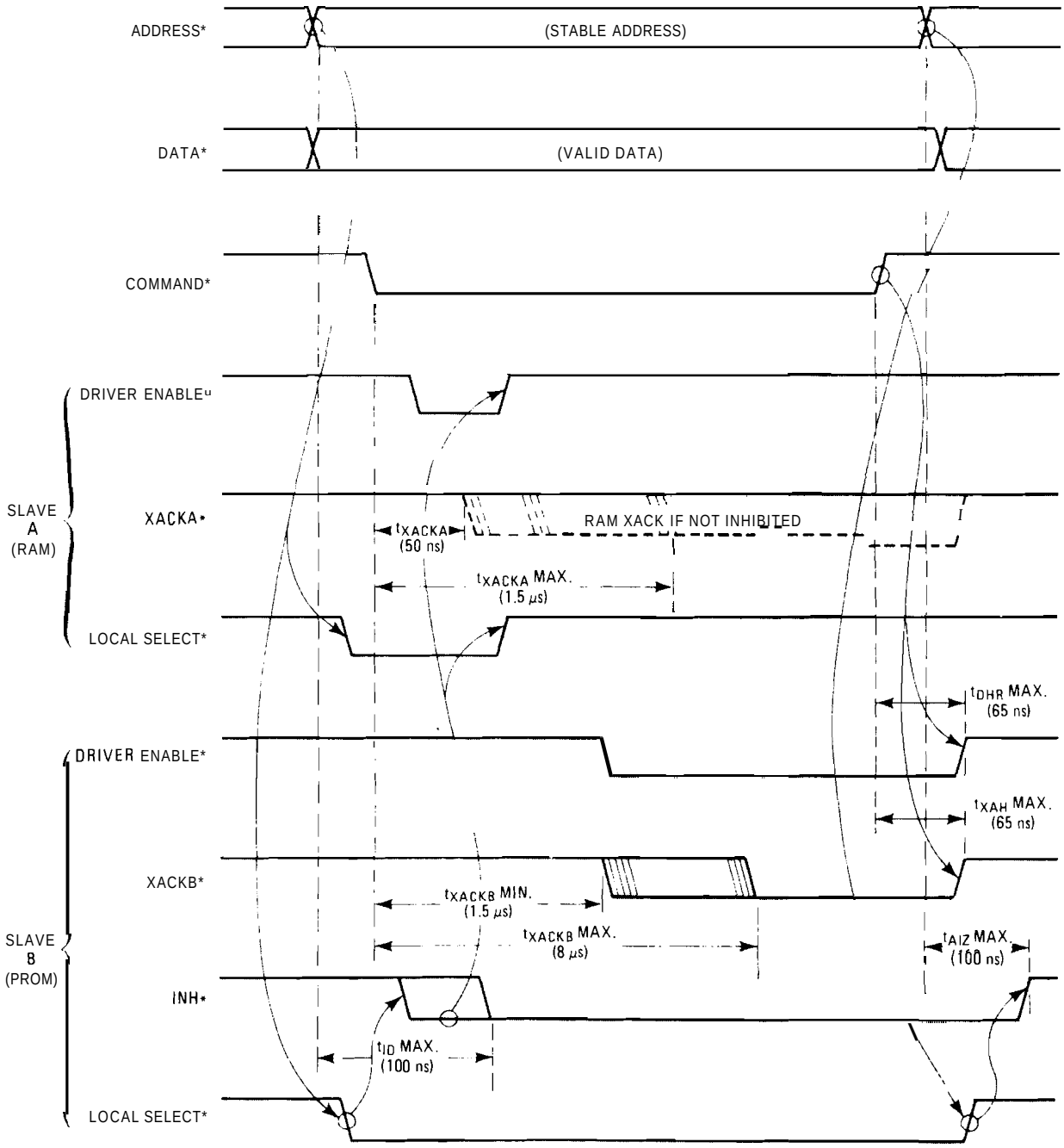


Fig 9
Inhibit Timing for Write Operation

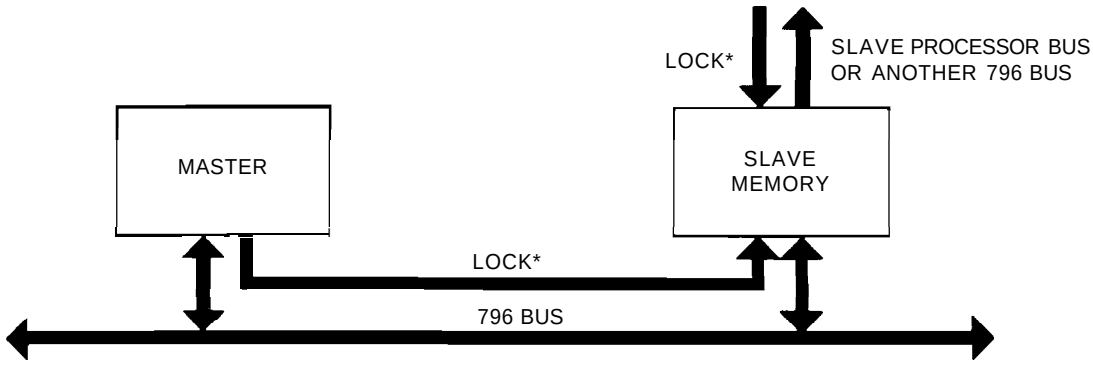


Fig 10
IEEE Std 796 Bus Lock Usage

is asserted during the appropriate address by a middle priority slave, such as a read-only memory module or memory-mapped I/O module, to inhibit the bus activity of a bottom priority slave, such as a read/write RAM module. INH2* is asserted at the appropriate address by a top priority slave, such as an auxiliary or bootstrap ROM module, to inhibit the bus activity of a middle priority slave. The top priority slave shall also assert INH1* so that a bottom priority slave will also be inhibited. The inhibit lines shall be asserted low by open collector or equivalent drivers. When both a middle and a top priority inhibiting slave are activated, INH1* will be asserted by drivers on both modules.

The use of the inhibit signals during memory reads (MRDC*) shall not cause any adverse effects within the inhibited slave module. That is, data in the inhibited slave shall not be altered and its status register, if any, shall not be affected.

The use of the inhibit signals during memory writes (MWTC*) shall be allowed, and might or might not affect the data within the inhibited slave. If the data is affected, it shall be only within the one byte (or word) that is being addressed. (No other data within the inhibited slave shall be altered.)

The inhibit signals, when issued, shall be generated within 100 ns (t_i) after the address is stable. (See Fig 9.) A command may be generated as early as 50 ns (t_{AS}) after the address is stable. This timing can cause the inhibit to occur after the command has been received by the inhibited module. To prevent false acknowledges, modules that can be inhibited should not generate an acknowledge until the inhibit signals have had time to become valid (50 ns after the command).

Figure 9 shows the timing for an inhibit operation. In this example, PROM and RAM have the same memory addresses; therefore, the PROM inhibits the RAM.

Although inhibit signals may be generated during IORC*, IOWC*, or INTA* operations, these signals are ignored by other slaves, including the slave that should respond to the INTA*, IORC*, or IOWC*.

2.2.2.8 Lock (LOCK*). The lock line is driven by the master control of the bus when a locked bus access is required. A locked access is typically required in a read-modify-write semaphore operation to prevent another processor from accessing the memory between the read and the write. The busy line allows for this mutual exclusion on the IEEE Std 796 bus. The lock line allows mutual exclusion to be extended off of the bus. The lock signal (LOCK*) shall be active 100 ns prior to the read or write command going away. It shall

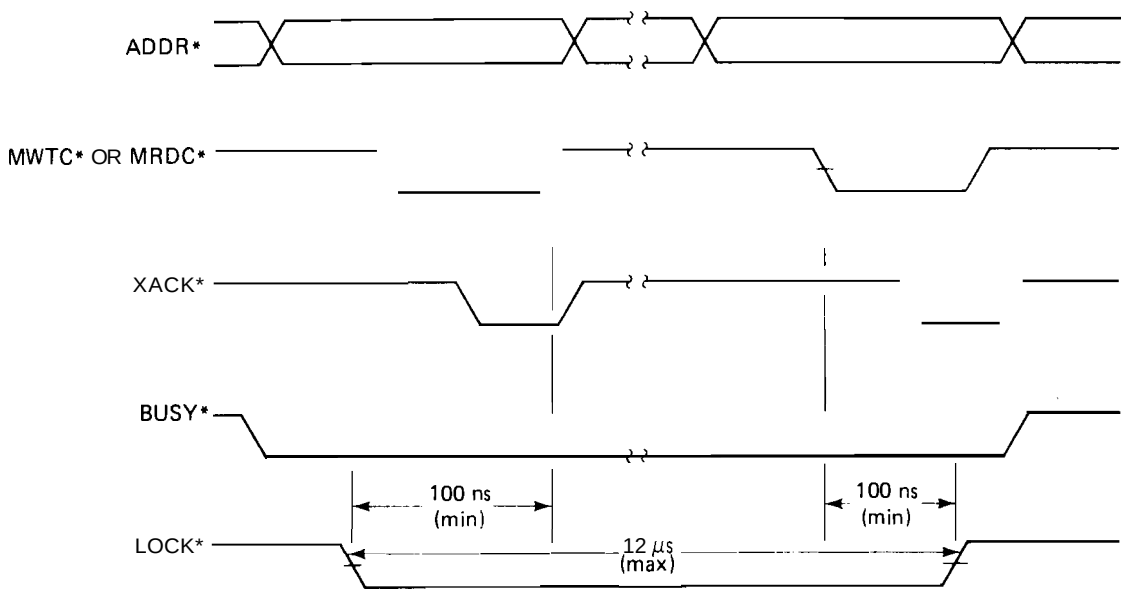


Fig 11
Lock Timing

remain active a minimum of 100 ns after the falling edge of the command signal for the last locked memory cycle. The slave locks its multiple ported memory to the IEEE Std 796 bus when it is addressed and the lock line is asserted. The lock signal should not be asserted for more than 12 μs continuously. This ensures the processor on the other side of the multiple ported memory that it will gain access to the memory in a reasonable amount of time. The busy signal (BUSY*) should be active whenever the lock line is asserted. Figure 11 shows the timing for the lock signal.

2.3 Interrupt Operations. The following subsections explain the IEEE Std 796 bus signal lines used in the interrupt operation and the two different types of interrupt implementation. See 5.1.4 for information on levels of compliance with respect to interrupt attributes.

2.3.1 Interrupt Signal Lines

2.3.1.1 Interrupt Request Lines (INTO*–INT7*). A set of interrupt request lines (INTO*–INT7*) is provided on the bus. An interrupt is generated by activating one of the eight interrupt request lines with an open-collector driver. All interrupts are level-triggered, rather than edge-triggered. Requiring no edge to trigger an interrupt allows several sources to be attached to each line. The interrupt request lines are prioritized, with INTO* having the highest priority and INT7* having the lowest priority.

2.3.1.2 Interrupt Acknowledge (INTA*). An interrupt acknowledge line (INTA*), driven by the bus master, requests the transfer of interrupt information on the bus. The specific information timed onto the bus depends on the implementation of the interrupt scheme. In general, the leading edge of INTA* indicates that the address bus is active; the trailing edge indicates that data is present on the data lines.

2.3.2 Classes of Interrupt Implementation. There are two types of interrupt implementation schemes: nonbus vectored (NBV) and bus vectored (BV). The two schemes are explained in the following subsections.

2.3.2.1 Nonbus Vectored Interrupts. Nonbus vectored (NBV) interrupts are those interrupts handled on the bus master that do not require the IEEE Std 796 bus for transfer of the interrupt vector address. The interrupt vector address is generated by the interrupt

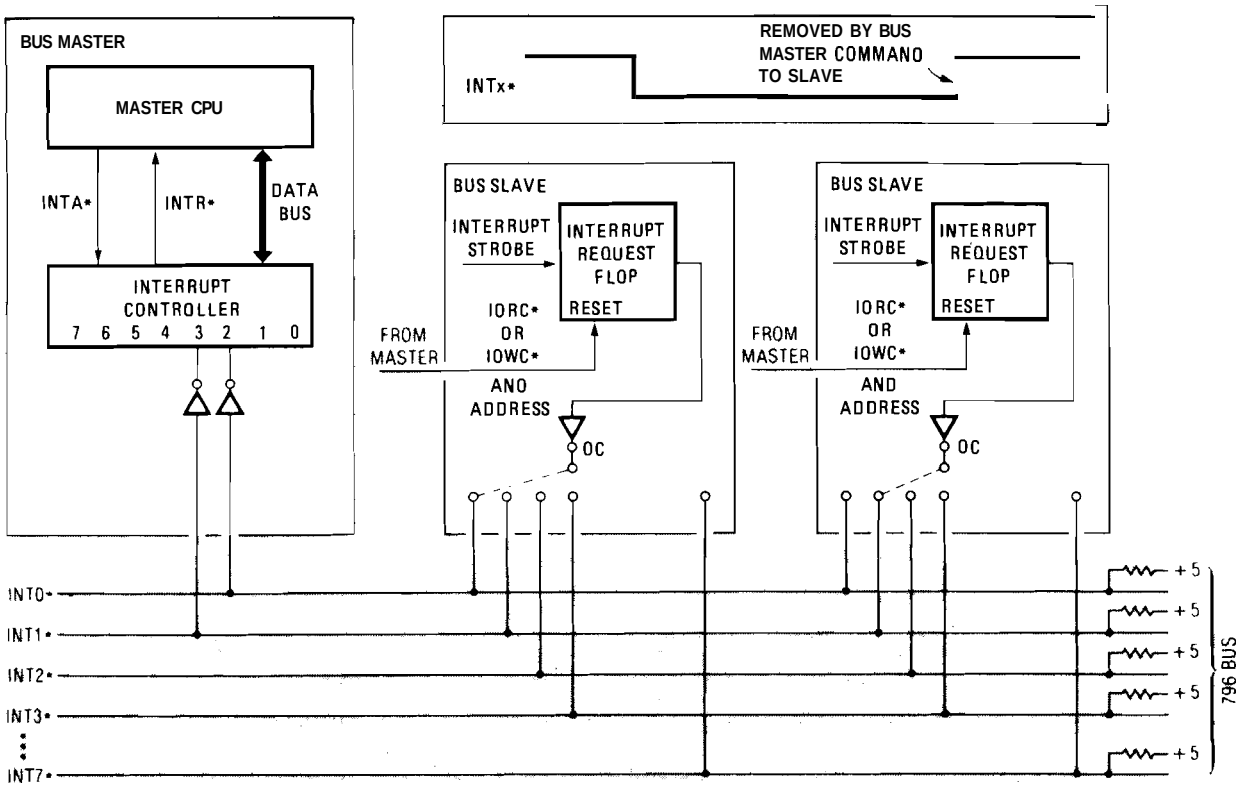


Fig 12
Non-Bus Vectored (NBV) Interrupt Logic

controller on the master and transferred to the processor over the local bus. The slave modules generating the interrupts can reside on the master module or on other bus modules, in which case they use the IEEE Std 796 bus interrupt request lines (INT0*–INT7*) to generate their interrupt requests to the bus master. When an interrupt request line is activated, the bus master performs its own interrupt operation and processes the interrupt. Figure 12 shows an example of NBV interrupt implementation.

2.3.2.2 Bus Vectored Interrupts. Bus vectored (BV) interrupts are those interrupts that transfer the interrupt vector address over the IEEE Std 796 bus from the slave to the bus master using the INTA* command signal.

When an interrupt request occurs, the interrupt control logic on the bus master interrupts its processor. The processor on the bus master generates the INTA* command, freezing the state of the interrupt logic for priority resolution. The bus master also overrides (retains the bus between bus cycles) the IEEE Std 796 bus to guarantee itself contiguous bus cycles. After the first INTA* command, the bus master's interrupt control logic puts an interrupt code on the IEEE Std 796 bus address lines. The interrupt code is the address of the highest priority active interrupt request line. At this point in the BV interrupt procedure, two different sequences can occur because the IEEE Std 796 bus can support masters that generate either two or three INTA* commands.

If the bus master generates two INTA* commands, one more INTA* command will be generated. This second INTA* causes the bus slave interrupt control logic to transmit its interrupt vector address on the IEEE Std 796 bus data lines. The address is used by the bus master to service the interrupt.

If the bus master generates three INTA* commands, two more INTA* commands will be generated. These two INTA* commands allow

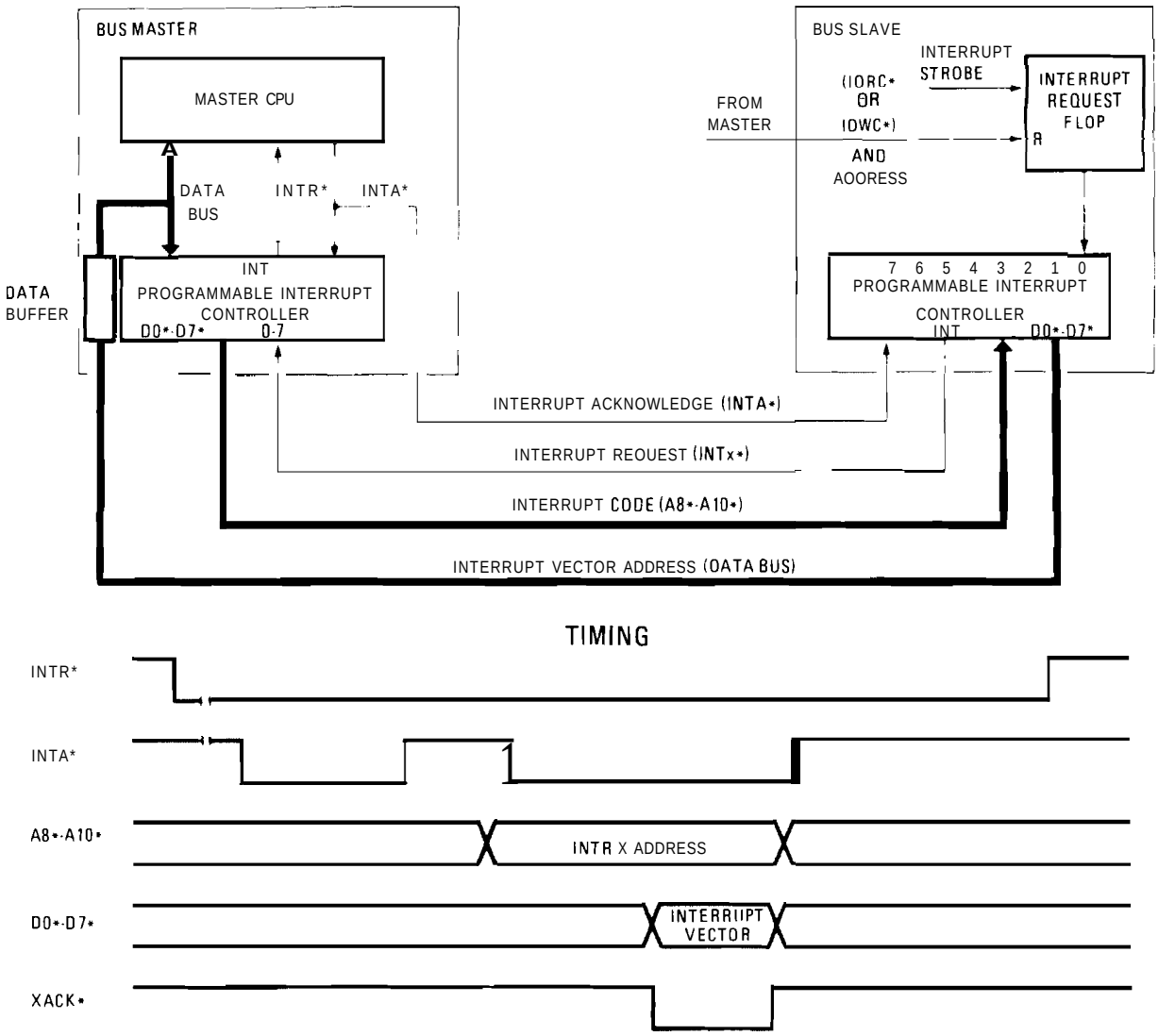


Fig 13
Bus Vectored (BV) Interrupt Logic

the bus slave to put its 2-byte interrupt vector address on the IEEE Std 796 bus data lines (one byte for each INTA*). The interrupt vector address is used by the bus master to service the interrupt.

NOTE: The IEEE Std 796 bus can support only one type of bus vectored interrupt in a given system. However, the IEEE Std 796 bus can support the bus vectored (BV) and nonbus vectored (NBV) interrupts within the same system.

Figure 13 depicts an example of BV interrupt implementation.

2.4 IEEE Std 796 Bus Exchange. The IEEE Std 796 bus can accommodate several bus masters on the same system, each taking control of the bus as it needs to effect data transfers. The bus masters request bus control through a bus exchange sequence.

The discussion of the IEEE Std 796 bus exchange will be separated into three parts. The first part explains the signals involved, the second part discusses the bus exchange priority techniques (serial and parallel), and the third part explains the implementation of the exchange logic.

2.4.1 IEEE Std 796 Bus Exchange Signals. A set of six signals is used to implement the bus exchange operation. All bus exchange signals are synchronized by BCLK*.

2.4.1.1 Bus Clock (BCLK*). This periodic clock signal is used to synchronize the exchange logic, with synchronization occurring on the trailing (high-to-low) edge of the pulse. BCLK* has a duty cycle of approximately 50%, a maximum frequency of 10 MHz, and can be slowed, stepped, or stopped as called for by system design. There is no requirement for synchronization between BCLK* and CCLK*, but they may be derived from the same source. The BCLK* line is driven by a TTL clock driver or equivalent.

2.4.1.2 Bus Busy (BUSY*). This signal is driven by the master in control of the bus. All other masters monitor BUSY* to determine the state of the bus. This bidirectional signal, which is driven by an open-collector gate, is synchronized by BCLK*.

2.4.1.3 Bus Priority In (BPRN*). A nonbused signal that indicates to a master that no master of higher priority is requesting control of the bus. BPRN* is synchronized by BCLK* and driven by TTL gates. In a serial resolution scheme, this is the master's input from the priority chain. In a parallel resolution scheme, this is the master's input from the parallel priority circuit.

2.4.1.4 Bus Priority Out (BPRO*). This nonbused signal, when activated by a bus master, indicates to the bus master of the next lower priority that it may gain control of the bus (that is, no higher priority requests are pending for control of the bus). This signal is used only in a daisy-chained serial priority resolution scheme and should be connected to the bus priority in (BPRN*) input of the next lower priority bus master. BPRO* is driven by TTL gates and is synchronized by BCLK*.

NOTE: Each bus master should allow its BPRO* signal to be disconnected from the BPRO* line on the IEEE Std 796 bus so that, if desired, a parallel priority resolution scheme can be used. This capability is to allow some bus masters to have their BPRN* inputs driven by a central parallel resolution circuit instead of by the BPRO* of the next higher priority master.

2.4.1.5 Bus Request (BREQ*). The bus request (BREQ*) line is used with the parallel priority resolution scheme and is a request of the master for IEEE Std 796 bus control. The priorities of the BREQ* from each master are resolved in a parallel priority resolution circuit. The highest priority request enables the BPRN* input of that master, allowing it to gain control of the bus. BREQ* is synchronized by BCLK* and is a TTL output.

2.4.1.6 Common Bus Request (CBRQ*) (Optional). Any master that wants control of the IEEE Std 796 bus but does not control it can activate CBRQ* with an opencollector gate. If CBRQ* is high, it indicates to the bus master that no other master is requesting the bus and therefore the present bus master can retain the bus. There are times when this can save the bus exchange overhead for the current master. This is because quite often when a master is controlling the bus, there are no other masters that are requesting the bus. Without CBRQ*, only BPRN* indicates whether or not another master is requesting the bus and, for BPRN*, only if the other master is of higher priority. Between the master's bus transfer cycles, so as to allow lower priority masters to take the bus if they need it, the master should give up the bus. At the start of the master's next transfer cycle, the bus should be regained. If no other master has the bus, this can take approximately three BCLK* periods. To avoid this overhead of unnecessarily giving up and regaining the bus when no other masters need it, CBRQ* may be used. Any master that wants but does not have the bus should drive this line low (true). The master that has the bus can, at the end of a transfer cycle, sense CBRQ*. If it is not low, then the bus does not have to be released, thereby eliminating the delay of regaining the bus at the start of the next cycle. (At any time before the master's next cycle, any other master desiring the bus will drive CBRQ* and cause the master to relinquish the bus at that time.)

Masters that use CBRQ* should be able to disable that function so that they can be used with masters that do not generate the CBRQ* signal.

2.4.2 Bus Exchange Priority Techniques. Two bus exchange priority techniques are discussed: a serial technique and a parallel technique. Fig-

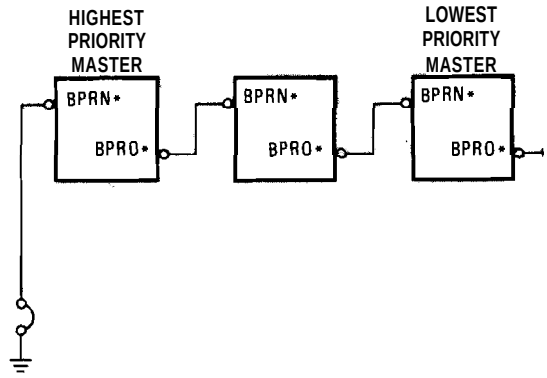


Fig 14
Serial Priority Technique

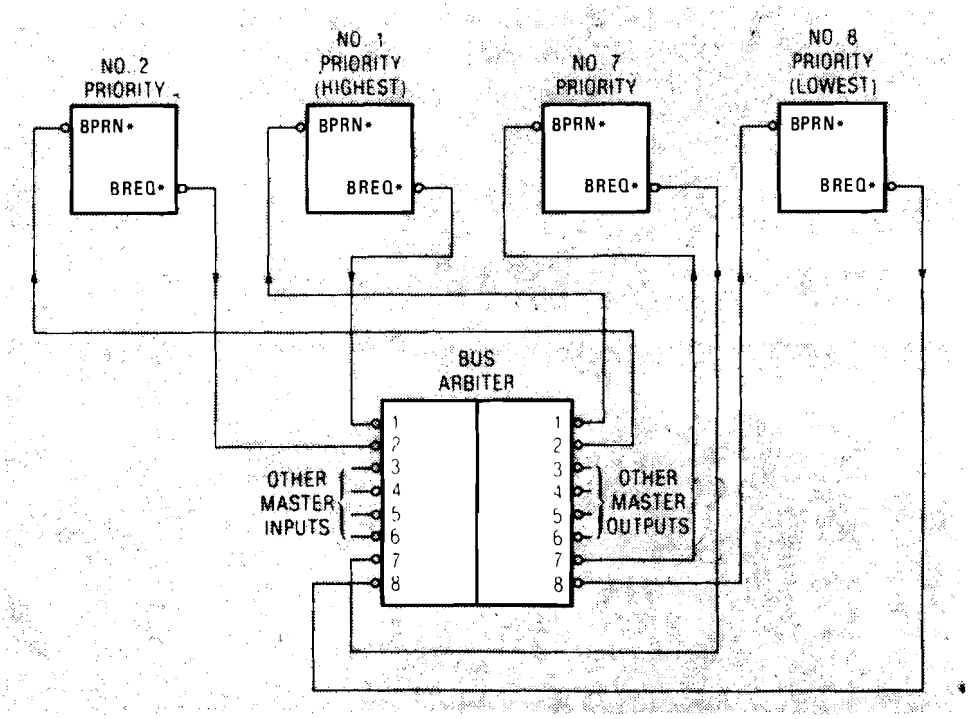


Fig 15
Parallel Priority Technique

ures 14 and 15 illustrate these two techniques. Note that the parallel and serial schemes are compatible and therefore can be combined and used together on the same bus. The bus exchange implementation discussed in 2.4.3 is the same for both techniques.

2.4.2.1 Serial Priority Technique. Serial priority resolution is accomplished with a daisy-chain technique (see Fig 14). With such a scheme, the bus priority output (BPRO*) of each master is connected to the bus priority input (BPRN*) of the next lower priority master. The BPRN* of the highest priority master in the serial chain shall either be always active or connected to a central bus arbiter as described in 2.4.2.2. The latter connection is used if a parallel-serial priority structure is used.

Serial priority resolution is accomplished in

the following manner. The BPRO* output for a particular master is asserted if and only if its BPRN* input is active and that a master is not requesting control of the bus. Thus, if a master requests control of the bus, it shall set its BPRO* high, which in turn disables the BPRN* of all lower priority masters. The number of masters that can be linked in a serial chain is limited to the fact that the BPRN* signal shall propagate through the entire chain within one BCLK* cycle. If the maximum BCLK* of 10 MHz is used, then the number of masters in a serial chain is limited to three.

2.4.2.2 Parallel Arbitration Technique. In the parallel technique, the bus allocation is determined by a bus arbiter (see Fig 15). This may be a priority scheme, which determines the next master by a fixed priority structure or some other mechanism for allocation (for example, sequential). The BREQ* lines are used by the arbiter to signal the next master on the appropriate BPRN* line. The BPRO* lines are not used in the parallel allocation BPRN* scheme.

3. Electrical Specifications

The electrical specifications for the IEEE Std 796 bus are as follows:

(1) General bus considerations of the state relationships, signal line characteristics, and power supplies

(2) Timing specifications for the bus signals

(3) Specifications for the signal line drivers and receivers, as well as the electrical termination requirements

When electrical specifications indicate minimum or maximum values for the bus, they should be measurable at any point on the bus.

Note that a particular implemented bus could have any amount of bus propagation delay and ringing (before setup times), as long as all bus parameters (for example, setup, hold, and other times) are met at all points on the bus. However, to facilitate the design of a compatible set of modules (masters and slaves) that use the bus, the standard maximum bus propagation delay will be specified as t_{PD} (maximum).

3.1 General Bus Considerations

3.1.1 Logical and Electrical State Relationships. The signal names indicate whether or not the signal lines on the IEEE Std 796 bus are active high or active low. If the signal name ends with a nathan ("*", a non-superscript asterisk), then the signal is active low and its logical-electrical state relationship for that signal is as shown in Table 3.

If the signal name has no nathan, then the signal is active high and its logical-electrical state relationship for that signal is as shown in Table 4.

These specifications are based on TTL, where the power source is 5 V ±5%, referenced to logic ground (GND).

When specified, current flow into a node has a positive sign and current flow out of a node has a negative sign.

3.1.2 Signal Line Characteristics. The following subsections describe two types of requirements. The first includes the requirements on the signal line that are measured when the signal line is in use. The second type includes those that are measured under special test conditions.

Table 3
Logical/Electrical State Relationship for "*" Signal

Logical State	Electrical Signal Level	At Receiver	At Driver
0	H=TTL high state	5.25 V ≥ H ≥ 2.0 V	5.25 V ≥ H ≥ 2.4 V
1	L=TTL low state	0.8 V ≥ L ≥ −0.5 V	0.5 V ≥ L ≥ 0 V

Table 4
Logical/Electrical State Relationship for Non "*" Signal

Logical State	Electrical Signal Level	At Receiver	At Driver
0	L=TTL low state	0.8 V ≥ L ≥ −0.5 V	0.5 V ≥ L ≥ 0 V
1	H=TTL high state	5.25 V ≥ H ≥ 2.0 V	5.25 V ≥ H ≥ 2.4 V

Table 5
Typical Rise and Fall Times

	Open Collector	Totem Pole	3-State
Rise time	—	10 ns	10 ns
Fall time	10 ns	10 ns	10 ns

3.1.2.1 In-Use Signal Line Requirements. During normal use, the rise and fall times of the signals depend on which type of driver is used (see 3.3). Typical rise and fall times are as shown in Table 5.

The typical signal propagation delay on the bus is t_{PD} (typ). This is measured from the edge of any one board plugged into the **IEEE** Std 796 bus to any other board plugged into the **IEEE** Std 796 bus:

t_{PD} (typ) = 3 ns

NOTE: For all boards plugged into the bus, the setup, hold, and any other times are measured at the edge of the board where it is plugged into the bus. This means that all board-internal delays should be taken into account, while still providing for the setup, hold, and other times.

After power-up, the following specifications apply:

(1) Bus termination required for each signal line (see 3.3)

(2) Settling time for all command line signals (see 2.2.2.5) after a transition is zero. On these lines the ringing cannot go beyond the noise immunity levels, that is, high, minimum; or low, maximum. This also applies to the transfer acknowledge and inhibit lines.

For all address lines (see 2.2.2.3) the signals shall be stable (settled) at least 50 ns before any command line goes active (setup time). This settling requirement means there can be no ringing beyond the noise immunity levels (high, minimum; low, maximum). These requirements also apply to the data lines (see 2.2.2.4) during any write operations.

For all data lines during read operations, the setup time is zero before the transfer acknowledge (XACK*) signal goes active; and the hold time is zero after the read-type command goes inactive.

The setup, hold, and command ringing are summarized and graphically presented in Fig 16.

Fig 16
Setup, Hold, and Command Ringing Summary

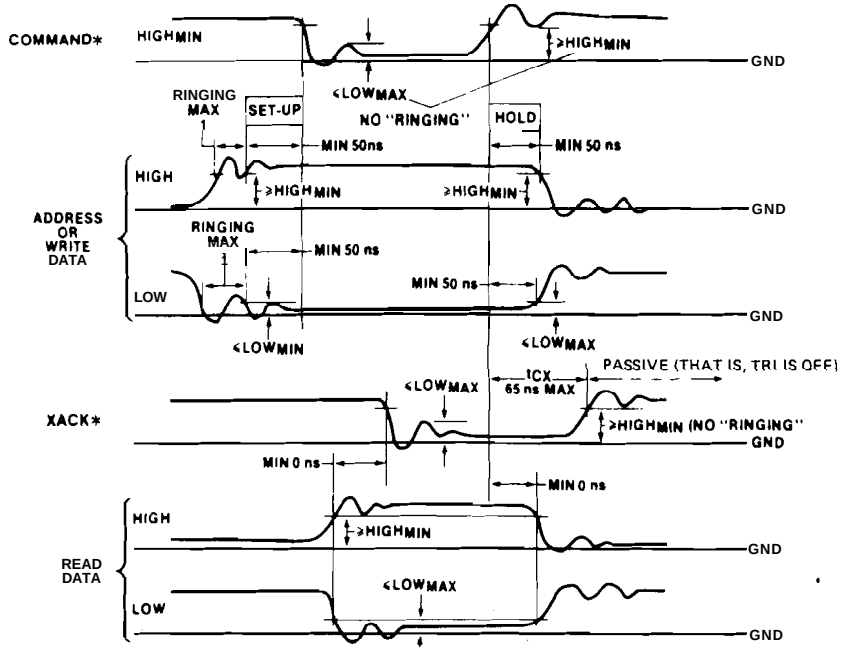


Table **6**
IEEE Std **796** Bus Power Supply Specifications

	Standard*			
Parameter	Ground	+5	+12	–12
Mnemonic	GND	+5 V	+12 V	–12 V
Bus pins	P1-1,2,11,12, 75,76,85,86	P1-3,4,5,6, 81,82,83, 84	P1-7,8	P1-79,80
Tolerance†	Ref	4.9–5.2	11.8–12.5	–12.5–(–11.8)
Combined line and load reg	Ref	1%	1%	1%
Ripple (peak to peak)**	Ref	50 mV	50 mV	50 mV
Transient response (50% load change)		500 μs	500 μs	500 μs

*Point of measurement is at connection point between motherboard and power supply.
†Includes line, load, temperature, and ripple effects.
**At 5 MHz bandwidth.

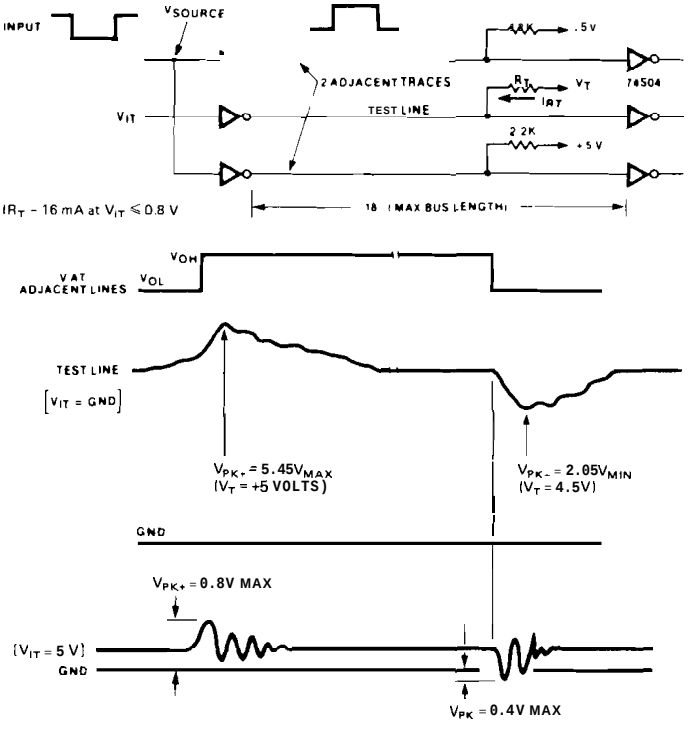


Fig **17**
Line-to-Line Coupling Characteristics

3.1.2.2 Backplane Signal Trace Characteristics. Requirements for line-to-line coupling characteristics are shown in Fig **17.** Thespecific test conditions under which the specifications are to be met are also shown.

3.1.3 Power Supply Specifications. Table **6** provides all power supply specifications. All voltages not shown in Table **6** that may be required on a board plugging into the IEEE Std **796** bus should be derived from one of the standard voltages **+5 V, +12 V, –12 V.**

3.1.4 Temperature and Humidity. Bus speci-

Table 7
IEEE Std 796 Bus Timing Specifications Summary

Parameter	Description	Minimum	Maximum	Units	Reference
t_{AH}	Address hold time	50		ns	3.2.1, 3.2.2, 3.2.4
t_{AIZ}	Address to inhibit high delay	0	100	ns	3.2.3
t_{AS}	Address setup time (at slave board)	50		ns	3.2.1, 3.2.2, 3.2.4
t_{BCY}	BCLK* Period	100	∞	ns	3.2.5
t_{BPRNO}	BPRN* to BPRO*	0	30	ns	3.2.5
t_{BPRNS}	BPRN* to ↓BCLK* setup time	22		ns	3.2.5
t_{BPRO}	↓BCLK* to BPRO*	0	40	ns	3.2.5
t_{BREQH}	↓BLCK* to BREQ* high delay	0	35	ns	3.2.5
t_{BREQL}	↓BCLK* to BREQ* low delay	0	35	ns	3.2.5
t_{BSYO}	CBRQ* and BUSY* to ↑BUSY	—	12	μs	3.2.5
t_{BUSY}	BUSY* delay from ↓BCLK*	0	70	ns	3.2.5
t_{BUSYS}	BUSY* to ↓BCLK setup time	25		ns	3.2.5
t_{BW}	BCLK* width	$0.35t_{BCY}$	$0.65t_{BCY}$		3.2.5
t_{CBRO}	↓BCLK* to CBRQ*	0	60	ns	3.2.5
t_{CBRQS}	CBRQ* to ↓BCLK* setup time	35		ns	3.2.5
t_{CCY}	CCLK* period	100	110	ns	3.2.6
t_{CMD}	Command pulse width	100	t_{TOUT}	ns	3.2.1, 3.2.2
t_{CMPH}	Command hold time	20		ns	3.2.1, 3.2.2
t_{CPM}	Central priority module Resolution delay (parallel priority)	0	$t_{BCY}-t_{BREQ}-2t_{PD}-t_{BPRNS}-t_{SKEW}$		3.2.5
t_{CSEP}	Command separation	100		ns	3.2.4, 3.2.6
t_{CW}	CCLK* width	$0.35t_{CCY}$	$0.65t_{CCY}$	ns	3.2.6
t_{DHR}	Read data hold time	0	65	ns	3.2.1, 3.2.4
t_{DHW}	Write data hold time	50		ns	3.2.2
t_{DS}	Write data setup time	50		ns	3.2.2
t_{DXL}	Read data setup time to XACK*	0		ns	3.2.1, 3.2.4
t_{IAD}	XACK* disable from inhibit (internal parameter on an inhibited slave; used to determine t_{XACKA} min)	0	100 (arbitrary)	ns	2.3.2
t_{ID}	Inhibit delay	0	100 (recommend <100 ns)	ns	3.2.3
t_{INIT}	INIT* width	5		ms	3.2.6, 3.2.7
t_{INTA}	INTA* width	250		ns	3.2.4
t_{LCKH}	LOCK* hold time from command active	100		ns	3.2.6
t_{LCKS}	LOCK* to command setup time	100		ns	3.2.6
t_{LOCK}	LOCK* width		12	μs	3.2.6
t_{OUT}	Timeout delay	1	dc (∞)	ms	—
t_{PD} (typ)	Standard bus propagation delay		3	ns	3.1.2, 3.2.5
t_{SKEW}	BCLK* skew		t_{PD}		3.2.5
t_{XACK}	XACK* time (for slaves without inhibit function	0	8	μs	3.2.1, 3.2.2, 3.2.4
t_{XACKA}	XACK* time of an inhibited slave	t_{IAD} +50 ns	1500	ns	3.2.3
t_{XACKB}	XACK* time of an inhibiting slave	1500	8000	ns	3.2.3
t_{XAH}	XACK* hold time	0	65	ns	3.2.1, 3.2.2, 3.2.4
Serial Priority	See 3.2.5				

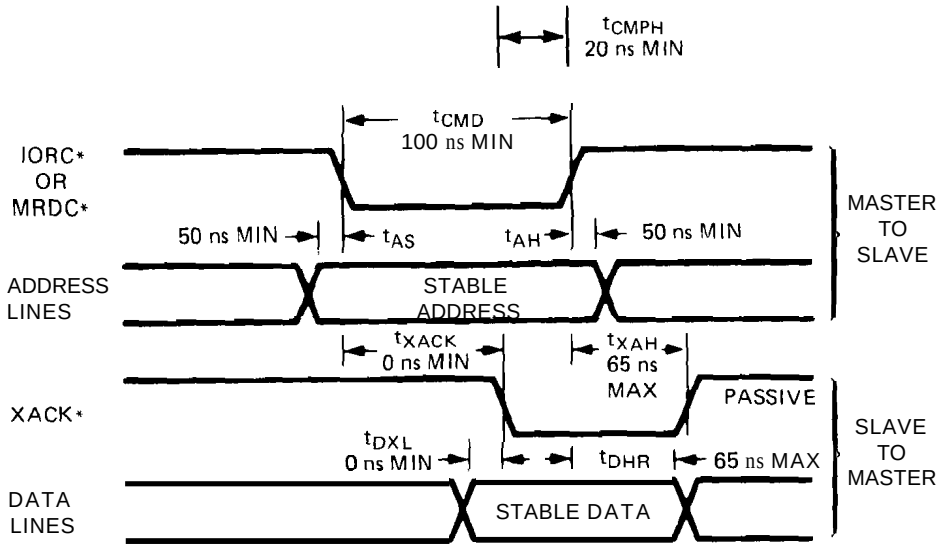


Fig 18
Read AC Timing

fications should be met with temperature and humidity within the following ranges:

Temperature: 0 °C – 55 °C (32 °F – 150 °F); freemoving air across modules and bus

Relative humidity: 90% maximum (no condensation)

This represents the standard environment for the IEEE Std 796 bus. It may be desirable to create more (or less) severe environmental restrictions in some applications.

3.2 Timing. This subsection describes all timing specifications on the IEEE Std 796 bus. It does not present descriptions or functional relationships (which are given in Section 2); however, this section does imply the functionality when relating two signals.

Table 7 summarizes the timing specifications in this section. For detailed descriptions, refer to the specific subsection(s) in the right-hand column.

The timing diagrams shown in this section usually show the minimum or maximum values required for each parameter. However, for clarity, the diagrams in this specification do not usually show both the minimum and maximum parameters. For this reason, the bus timing specification (Table 7) should be referenced for complete information. The timing diagrams show how all of the parameters are defined in relation to the signals involved. All timing is measured at 1.5 V with loading capacitance of C_o and terminations as specified in Table 8.

3.2.1 Read Operations (I/O and Memory). A read operation transfers data from memory or from I/O to the master that is controlling the bus (see 2.2). The lines involved and timing specifications for a read operation are shown in Fig 18. See the special inhibit operation in 3.2.3.

Table 8
Bus Drivers, Receivers, and Terminations

Bus Signals	Driver†§ Location	Type	I_{OL} Min mA	I_{OH} Min µs	I_{OH} Min µs	C_O Min pf	Receiver* Location	I_{IL} Max mA	I_{IH} Max µs	C_I Max pf	Termination Location	Type	R	Units
DAT0*–DATF* (16 lines)	Masters and slaves	TRI	16	–2000	—	300	Masters and slaves	–0.8	125	18	1 place	Pullup	2.2	kΩ
A0*–A19*, BHEN* (25 lines)	Masters	TRI	16	–2000	—	300	Slaves	–0.8	125	18	1 place	Pullup	2.2	kΩ
MRDC*, MWTC*	Masters	TRI	32	–2000	—	300	Slaves (memory; memory-mapped I/O)	–2	125	18	1 place	Pullup	1	kΩ
IORC*, IOWC*	Masters	TRI	32	–2000	—	300	Slaves (I/O)	–2	125	18	1 place	Pullup	1	kΩ
XACK*	Slaves	TRI	32	–400	—	300	Masters	–2	125	18	1 place	Pullup	510	Ω
INH1*, INH2*	Inhibiting slaves	OC	16	—	250	300	Inhibited slaves (RAM, PROM, ROM, memory-mapped I/O)	–2	50	18	1 place	Pullup	1	kΩ
BCLK*	1 place (master usually)	TTL	48	–3000	—	300	Masters	–2	125	18	Motherboard	To +5 V To GND	220 330	Ω Ω
BREQ*	Each master	TTL	10	–200	—	60	Central priority module	–2	50	18	Central priority module (not req)	Pullup	1	kΩ
BPRO*	Each master	TTL	3.2	–200	—	60	Next master in serial priority chain at its BPRN*	–1.6	50	18	(not req)			
BPRN*	Parallel: central priority module Serial: previous master's BPRO*	TTL	16	–400	—	300	Masters	–4	100	18	(not req)			
LOCK*	Master	TRI	32	–2000	—	300	All	–2	125	18	1 place	Pullup	1	kΩ
BUSY*, CBRQ*	All masters	OC	20	—	250	300	All masters	–2	50	18	1 place	Pullup	1	krz
INIT*	Master	OC	32	—	250	300	All	–2	50	18	1 place	Pullup	2.2	kΩ
CCLK*	1 place	TTL	48	–3000	—	300	Any	–2	125	18	Motherboard	To +5 V To GND	220 330	Ω Ω
INTA*	Masters	TRI	32	–2000	—	300	Slaves (interrupting I/O)	–2	125	18	1 place	Pullup	1	kΩ
INTO*–INT7* (8 lines)	Slaves	OC	16	—	250	300	Masters	–1.6	40	18	1 place	Pullup	1	kΩ

†Driver requirements:
I_{OH} = High output current drive
I_{OL} = Low output current drive
C_O = Capacitance drive capability
TRI = 3-State drive
OC = Open collector driver
TTL = Totem-pole driver

‡Receiver requirements:
I_{IH} = High input current load
I_{IL} = Low input current load
C_I = Capacitive load

§ For low- and high-voltage specifications see 3.1.1.

§§ ±5%, ¼W resistors

††All termination resistors specified as "1 place" are typically located on the motherboard.

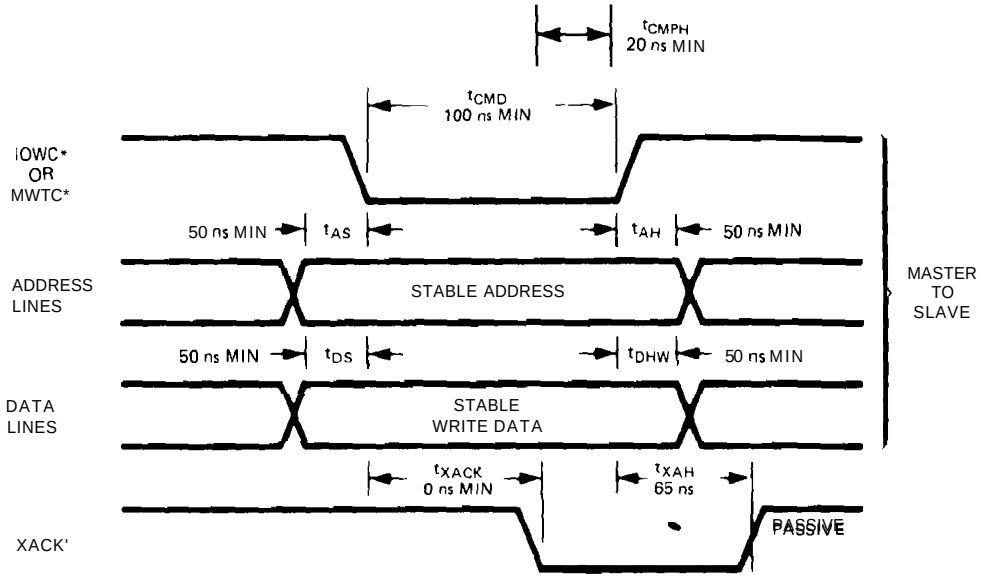


Fig 19
Write AC Timing

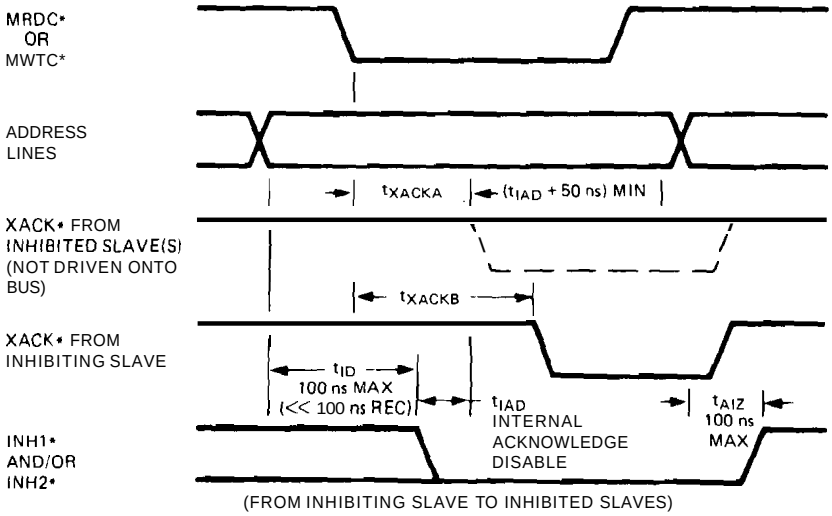


Fig 20
Inhibit AC Timing

3.2.2 Write Operations (I/O and Memory). A write operation transfers data from the master controlling the bus to memory or I/O (see 2.2). Timing for a write operation is shown in Fig 19. See 3.2.3 for inhibit operations.

3.2.3 Inhibit Operations. An inhibit operation may accompany any memory read or memory write operation. The main effect is for one slave to inhibit another slave from driving the data lines and from returning (driving) an acknowledge (XACK*). I/O addresses cannot be inhibited. Although inhibit signals may be generated during IORC*, IOWC*, or INTA* operations these signals are ignored by other slaves (including the slave that should respond to the INTA*, IORC*, or IOWC*). Inhibit timing is as illustrated in Fig 20. Related subsections are as follows:

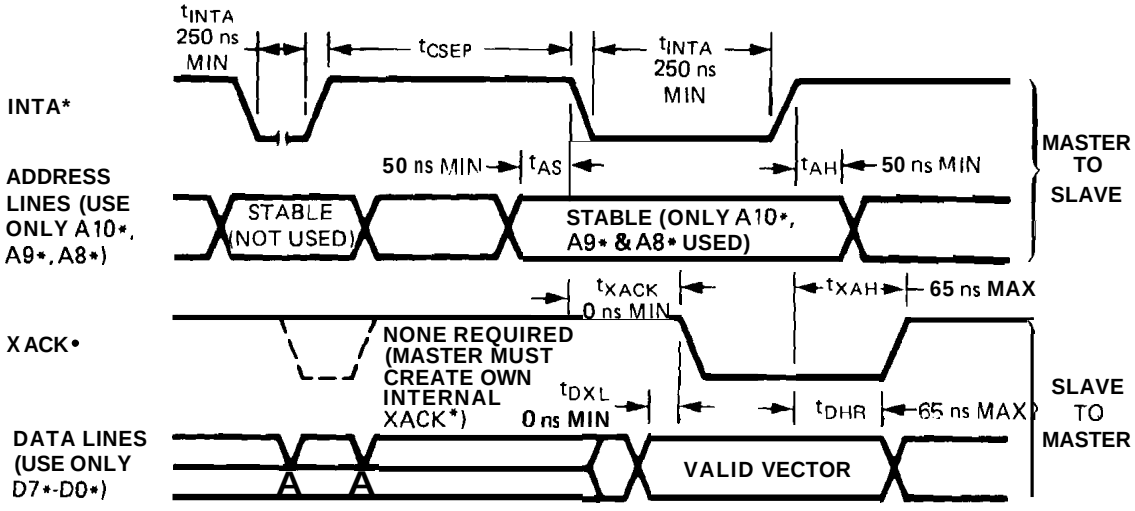
2.1.3.2.3	Functional descriptions
3.2	Timing specification summary
3.2.1	Read operations
3.2.2	Write operations
3.2.4	Interrupt implementations

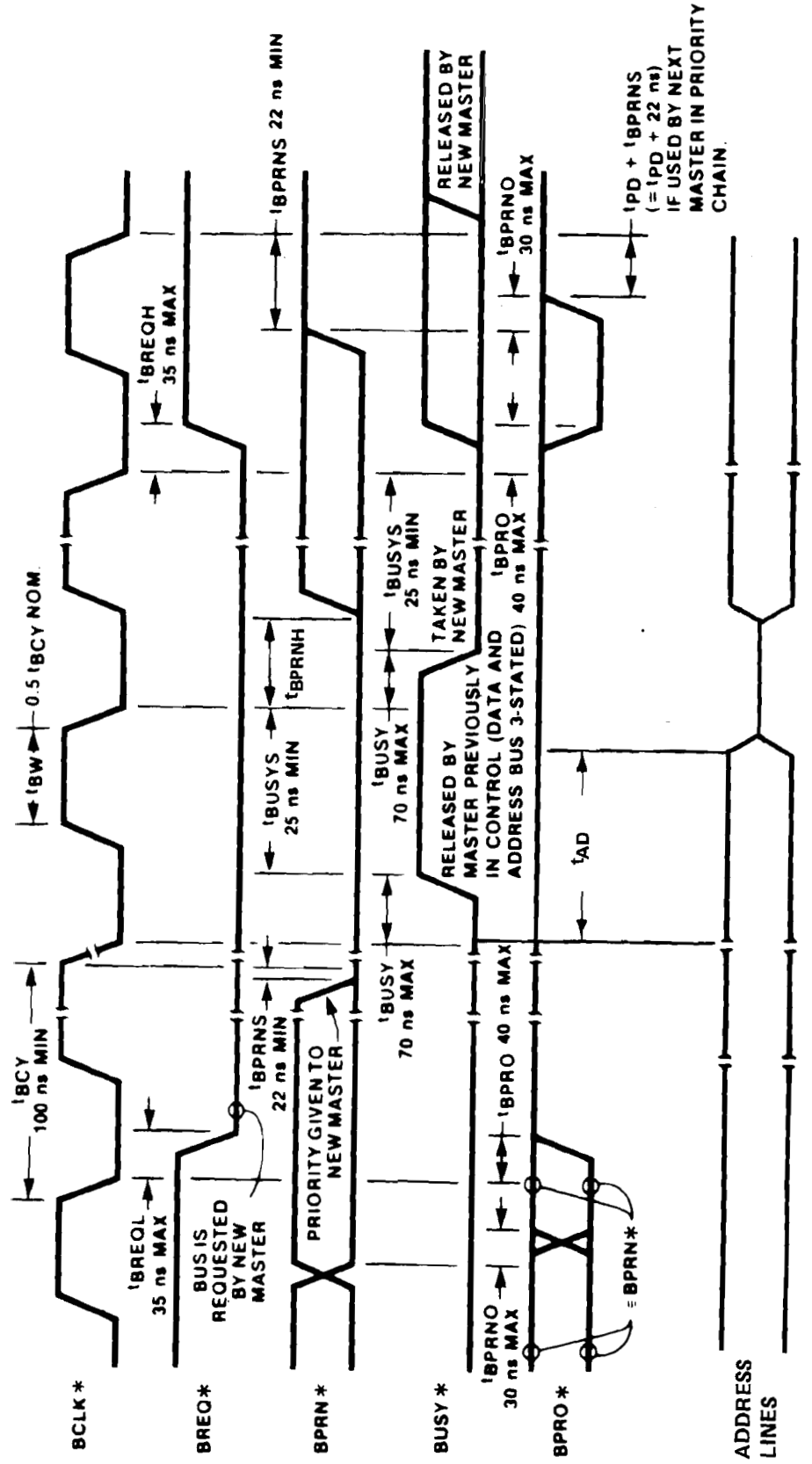
3.2.4 Interrupt Implementations. There are two types of interrupt implementation schemes: Non-bus vectored (NBV) and bus vectored (BV).

3.2.4.1 NBV Interrupts. NBV interrupts are handled on the bus master and do not require the IEEE Std 796 bus for transfer of an interrupt vector address. The slave modules generating the interrupts may reside on the master module or on other bus modules, in which case they use the IEEE Std 796 bus interrupt request lines (INT0*–INT7*) to generate interrupt requests to the bus master. When an interrupt request line is activated, the bus master performs its own internal interrupt operations and then processes the interrupt.

3.2.4.2 BV Interrupts. BV interrupts are those interrupts that transfer the interrupt vector address along the IEEE Std 796 bus from the slave to the bus master in response to the INTA* command signal. BV interrupt timing is shown in Fig 21.

Fig 21
Bus Vectored (BV) Interrupt AC Timing





NOTE: USE t_{PD}, BUS PROPAGATION DELAY, IN ALL SYSTEM CALCULATIONS

Fig 22
Bus Exchange AC Timing

When an interrupt request occurs, the interrupt control logic on the bus master interrupts its processor. The processor on the bus master generates an INTA* command, which freezes the state of the interrupt logic on the IEEE Std **796** bus for priority resolution. The bus master also overrides the IEEE Std **796** bus (retainsthe bus between bus cycles) to guarantee itself back-to-back bus cycles. After the first INTA* command, the bus master's interrupt control logic puts an interrupt code onto the IEEE Std **796** bus address lines. The interrupt code is the address of the highest priority active interrupt request line. At this point in the BV interrupt procedure, two different sequences could take place. The difference occurs because the IEEE Std 796 bus can support masters that generate either two or three INTA* commands during the interrupt process.

If the bus master generates two INTA* commands, one more INTA* command will be generated. This second INTA* causes the bus slave interrupt control logic to transmit its interrupt vector address on the IEEE Std **796** bus data lines. The address is used by the bus master to service the interrupt.

If the bus master generates three INTA* commands, two more INTA* commands will be generated. These two INTA* commands allow the bus slave to put its 2-byte interrupt vector address onto the data lines (one byte for each INTA* command). The interrupt vector address is used by the bus master to service the interrupt.

NOTE: The IEEE Std 796 bus can support only one type of BV interrupt in a given system. However, it can support both BV and NBV interrupts in the same system.

Subsections related to BV and NBV interrupts are:

2.3.2.2 Functional descriptions
3.2 Timing specification summary

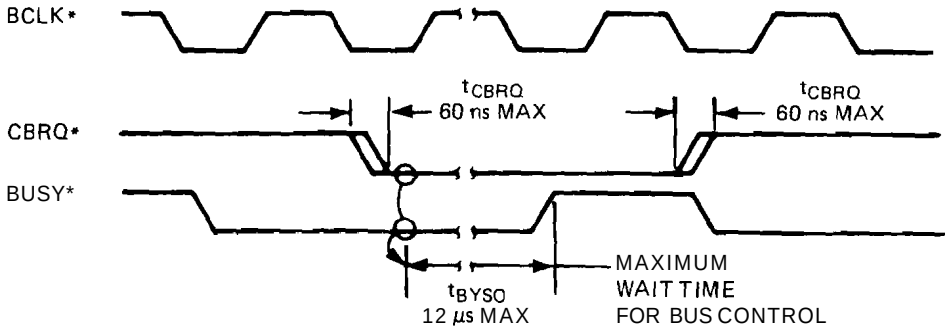
3.2.5 Bus Control Exchanges. A bus control exchange takes control of the bus (that is, the ability to do read, write, and interrupt acknowledge operations) from one master and gives it to another master. See 2.4 for a functional description of this process.

For a master that does not use the bus signal CBRQ* (common bus request), the timing specifications in Fig 22 apply.

For a system using CBRQ* (common bus request), each master must also satisfy the timing requirements illustrated in Fig 23. Note that before ***releasing the bus*** (that is, releasing BUSY*), the hold times, etc, of any ending cycle should still be met as described in the previous subsections of Section **3.** Likewise, after ***taking the bus*** (that is, driving BUSY* low), it is necessary to satisfy all applicable setup and other timing parameters for a cycle just beginning.

3.2.5.1 Serial Priority. For a system that uses a serial priority scheme (that is, daisy-chain BPRO*s to BPRN*s) the timing specifications in Fig 24 apply (see 2.4).

Fig 23
Common Bus Request AC Timing



(USE t_{PD}, BUS PROP DELAY, IN ALL SYS. CALC'S)

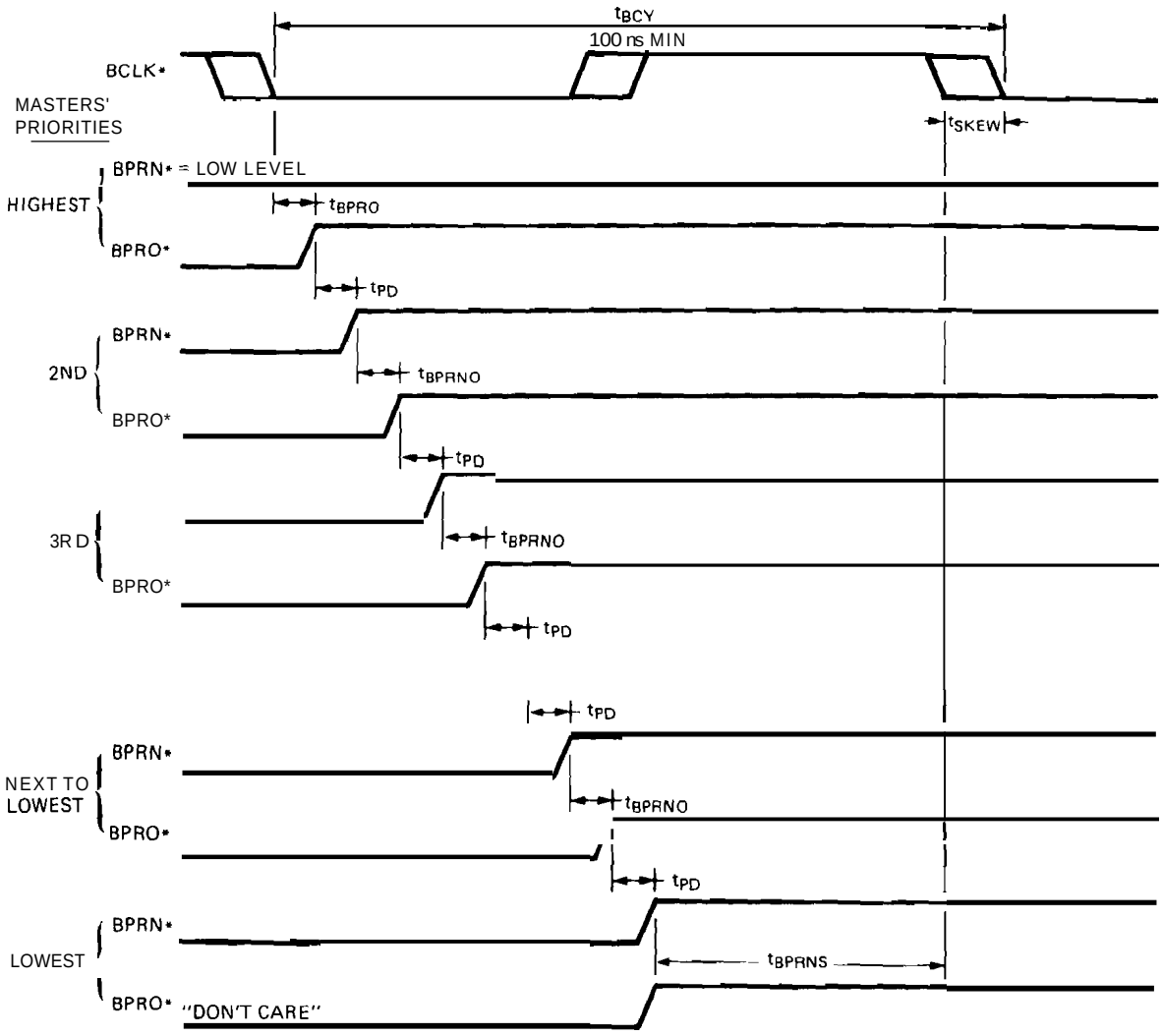
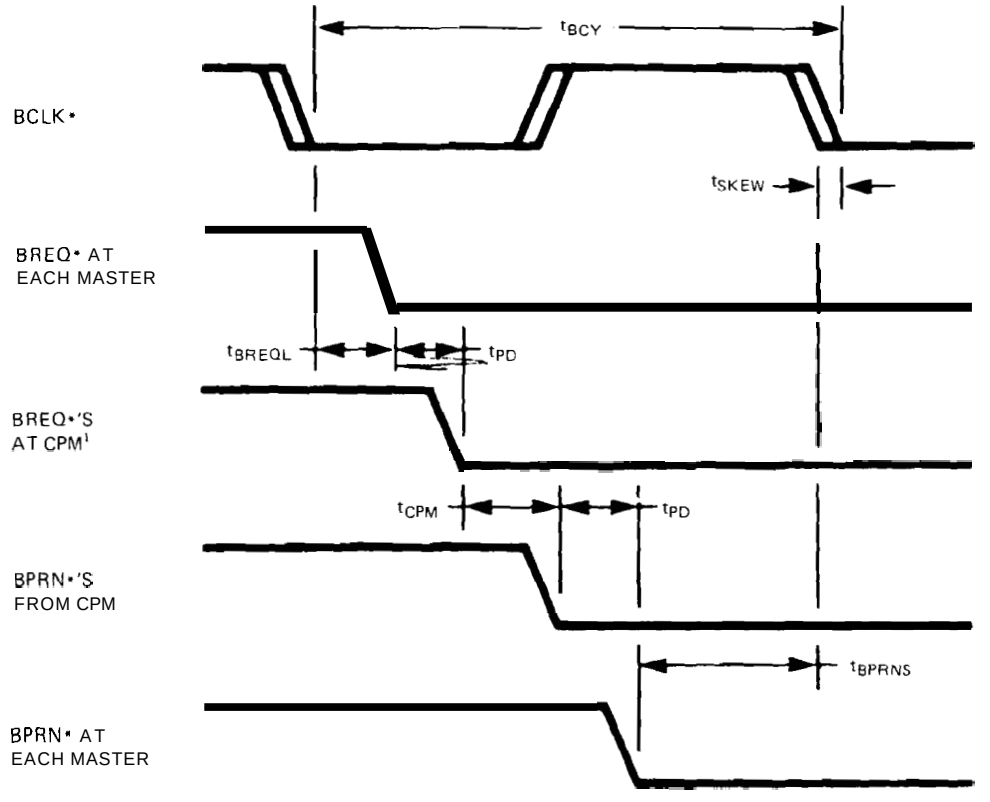


Fig 24
Serial Priority AC Timing

3.2.5.2 Parallel Priority. For a system that uses a parallel priority scheme (that is, a central priority resolver) the following system and CPM, central priority module, timing specifications of Fig 25 apply (see 2.4).

3.2.6 Miscellaneous Timing. The timing diagrams in Figs 26, 27, 28, and 29 show the timing of constant clock (CCLK*), command separation (t_{CSEP}), initialize (t_{INIT}), and lock (LOCK*), respectively.

3.3 Receiver Modules, Driver Modules and Terminations. Nontiming specifications unique to each signal line or to groups of signal lines are presented in Table 8. The requirements for the signal line receivers, drivers,,and bus terminations and the locations of the receiver, driver, and termination for each signal are given.



SPEC:

$t_{CPM(MAX)} \leq t_{BCY_{MIN}}\ t_{BREQ_{MAX}}\ 2t_{PD_{MAX}}\ t_{BPRNS_{MAX}}\ t_{SKEW_{MAX}}$

[1]CPM IS CENTRAL PRIORITY MODULE

Fig 25
Parallel Priority AC Timing

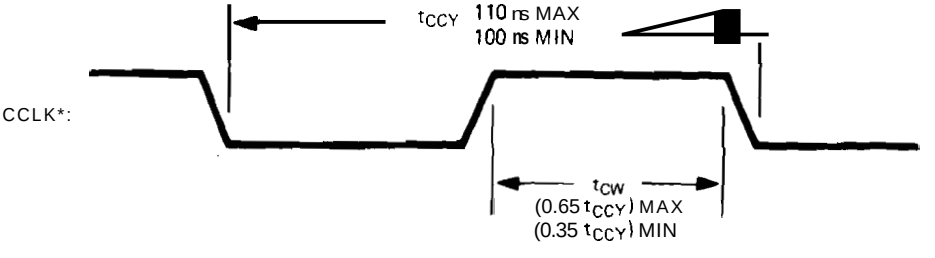


Fig 26
Constant Clock AC Timing

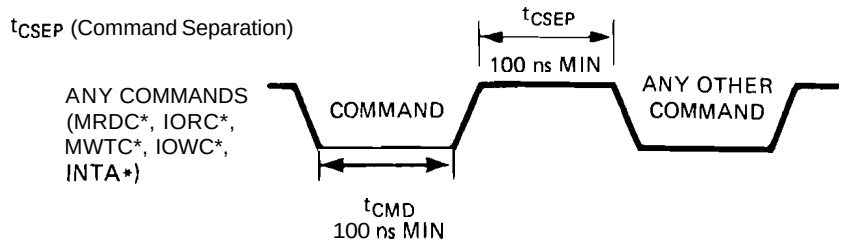


Fig 27
Command Separation AC Timing

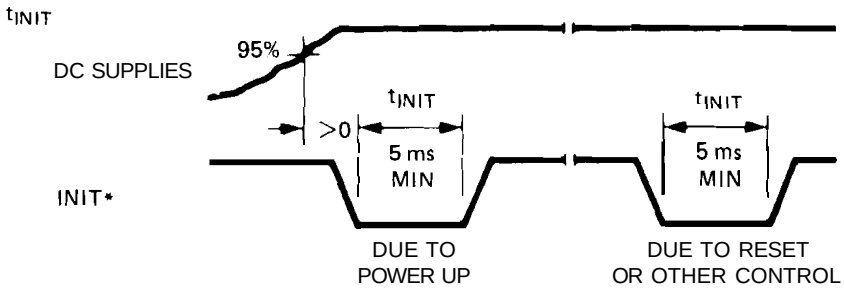


Fig 28
Initialize AC Timing

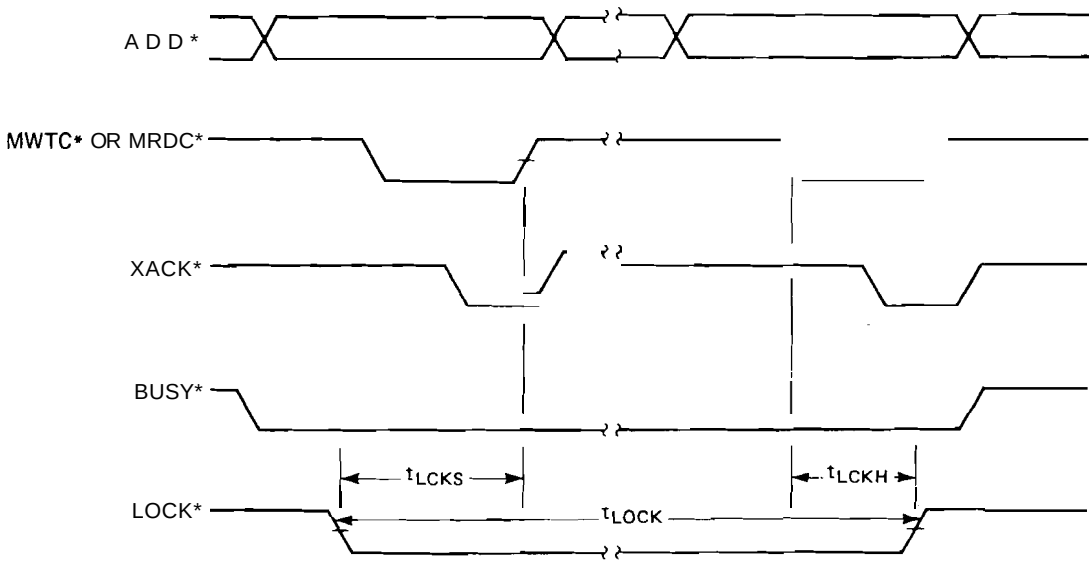


Fig 29
Lock AC Timing

4. Mechanical Specifications

This section describes all the physical and mechanical specifications that a designer must be concerned with when designing an IEEE Std 796 bus backplane or when designing printed circuit boards that will plug into the IEEE Std 796 bus interface. All dimensions are in inches with millimeters added in parentheses for reference only. Inch dimensions govern. See Fig 30.

4.1 Backplane Considerations. This section is a discussion of the things that the designer must consider when designing an IEEE Std 796 bus backplane.

The maximum length of the backplane connecting modules is 18 in (457.2 mm). Extender boards used within the system will not be supported by the bus unless the overall resulting length of the bus including the extender card is less than the 18 in maximum.

4.1.1 Board to Board Relationships. The following printed circuit board specifications shall be adhered to when designing IEEE Std 796 bus compatible boards which are to operate in an 0.6 inch board to board spacing backplane.

(1) ***Board to Board Spacing (L_C).*** Center to center of boards when plugged into backplane must be at least 0.6 ±0.02 in (15.24 ±0.51 mm).

(2) ***Roard Thickness (L,).*** The typical board thickness is 0.062 ±0.005 in (1.575 ±0.128 mm).

(3) ***Component Lead Length (L_L).*** The length of the component leads below the printed circuit board cannot exceed 0.093 in (2.362 mm).

(4) ***Component Height (L_H).*** The following equation is used to determine the maximum height of the components above the printed circuit board

$L_H < L_C - L_T - L_L$

$L_H < 0.58$ in − 0.067 in − 0.093 in

(14.73 mm − 1.702 mm −2.362 mm)

$L_H < 0.420$ in (10.67 mm) (including board warpage)

Fig 30
IEEE Std 796 Bus Backplane Card to Card Separation

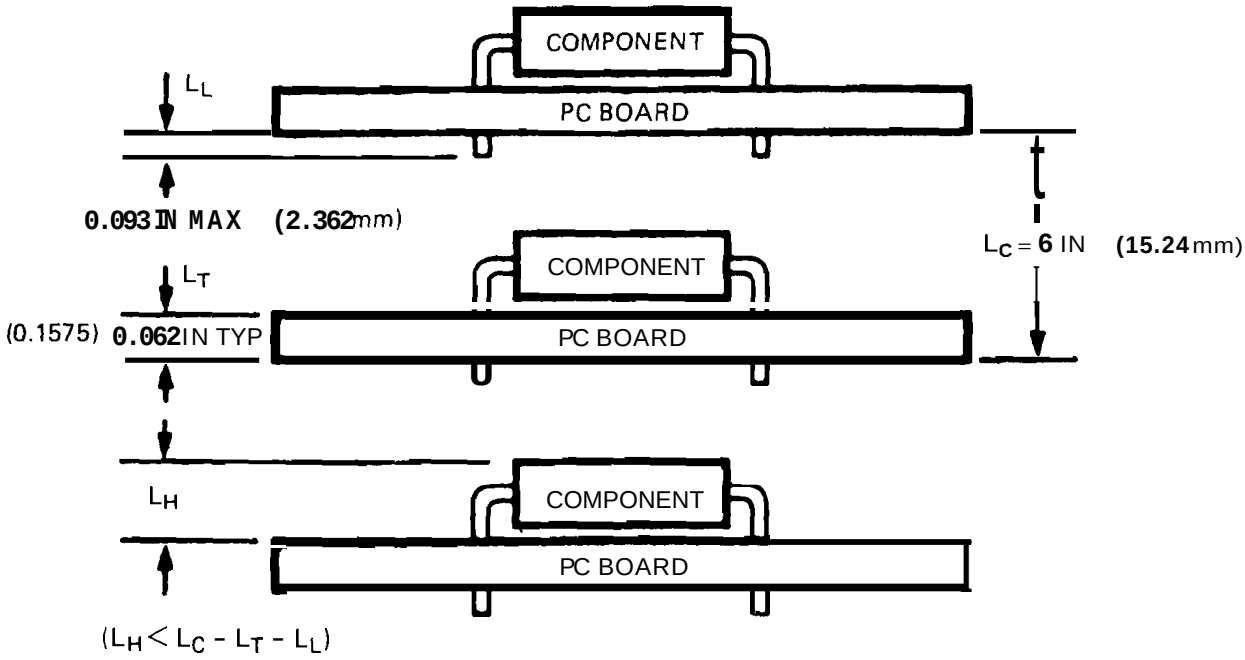
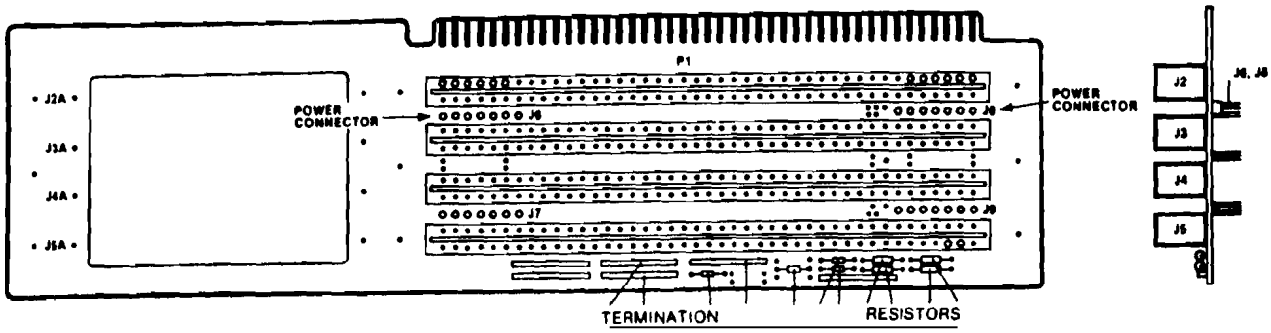


Table 9
Pin Assignment of Bus Signals on IEEE Std 796 Bus Board Connector (P1)

	(Component Side)			(Circuit Side)		
	Pin	Mnemonic	Description	Pin	Mnemonic	Description
Power supplies	1	GND	Signal GND	2	GND	Signal GND
	3	+5 V	+5 V dc	4	+5 V	+5 V dc
	5	+5 V	+5 V dc	6	+5 V	+5 V dc
	7	+12 V	+12 V dc	8	+12 V	+12 V dc
	9		Reserved, bused	10		Reserved, bused
	11	GND	Signal GND	12	GND	Signal GND
Bus controls	13	BCLK*	Bus clock	14	INIT*	Initialize
	15	BPRN*	Bus priority in	16	BPRO*	Bus priority out
	17	BUSY*	Bus busy	18	BREQ*	Bus request
	19	MRDC*	Memory read command	20	MWTC*	Memory write command
	21	IORC*	I/O read command	22	IOWC*	I/O write command
	23	XACK*	XFER acknowledge	24	INH1*	Inhibit 1 (disable RAM)
Bus controls and address	25	LOCK*	Lock	26	INH2*	Inhibit 2 (disable PROM or ROM)
	27	BHEN*	Byte high enable	28	AD10*	
	29	CBRQ*	Common bus request	30	AD11*	Address bus
	31	CCLK*	Constant clock	32	AD12*	
	33	INTA*	Interrupt acknowledge	34	AD13*	
Interrupts	35	INT6*	Parallel interrupt requests	36	INT7*	Parallel interrupt requests
	37	INT4*		38	INT5*	
	39	INT2*		40	INT3*	
	41	INTO*		42	INT1*	
Address	43	A14* (ADRE*)	Address bus	44	A15* (ADRF*)	Address bus
	45	A12* (ADRC*)	(See NOTE)	46	A13* (ADRD*)	(See NOTE)
	47	A10* (ADRA*)		48	A11* (ADRB*)	
	49	A8* (ADR8*)		50	A9* (ADR9*)	
	51	A6* (ADR6*)		52	A7* (ADR7*)	
	53	A4* (ADR4*)		54	A5* (ADR5*)	
	55	A2* (ADR2*)		56	A3* (ADR3*)	
	57	AO* (ADRO*)		58	A1* (ADR1*)	
Data	59	D14* (DATE*)	Data bus	60	D15* (DATF*)	Data bus
	61	D12* (DATC*)	(See NOTE)	62	D13* (DATD*)	(See NOTE)
	63	D10* (DATA*)		64	D11* (DATB*)	
	65	D8* (DAT8*)		66	D9* (DAT9*)	
	67	D6* (DAT6*)		68	D7* (DAT7*)	
	69	D4* (DAT4*)		70	D5* (DAT5*)	
	71	D2* (DAT2*)		72	D3* (DAT3*)	
	73	DO* (DATO*)		74	D1* (DAT1*)	
Power supplies	75	GND	Signal GND	76	GND	Signal GND
	77		Reserved, bused	78		Reserved, bused
	79	−12 V	−12 V dc	80	−12 V	−12 V dc
	81	+5 V	+5 V dc	82	+5 V	+5 V dc
	83	+5 V	+5 V dc	84	+5 V	+5 V dc
	85	GND	Signal GND	86	GND	Signal GND

NOTE: All Reserved pins are reserved for future use and should not be used if upwards compatibility is desired. Address and Data Mnemonics in parenthesis are for historical reference.



PARTS LIST

1 PWS TERMINATION BACKPLANE
2 7 POST WAFER CONNECTORS (.156" PIN CENTERS) (J6 AND J8)
4 EDGE BOARD CONNECTORS, 43/86 PINS ON .156" CENTERS (J2-J5)
12 WIRE WRAP POSTS
4 10 PIN, 2.2K, 9 RES, 1.5W RESISTOR PACKS (RP1-RP4)
1 10 PIN, 1K, 9 RES, 1.5W RESISTOR PACK (RP5)
1 10 PIN. 1.1K, 9 RES, 1.5W RESISTOR PACK (RP6)
1 1K RESISTOR, 1/8W, +5% (R1)
1 2.2K RESISTOR, 1/8W, ±% (R5)
2 220Ω RESISTORS, 1/4W, ±5% (R9, R11)
2 330Ω RESISTORS, 1/4W, ±5% (R10, R12)
2 510Ω RESISTORS, 1/8W, ±5% (R7, R8)

Fig 31
Typical IEEE Std 796 Bus Backplane

Table 10
Pin Assignment of Bus Signals on IEEE Std 796 Bus Board Connector (P2)

	(Component Side)			(Circuit Side)		
	Pin	Mnemonic	Description	Pin	Menmonic	Description
	1		Reserved, not bused	2		Reserved, not bused
	3		Reserved, not bused	4		Reserved, not bused
	5		Reserved, not bused	6		Reserved, not bused
	7		Reserved, not bused	8		Reserved, not bused
	9		Reserved, not bused	10		Reserved, not bused
	11		Reserved, not bused	12		Reserved, not bused
	13		Reserved, not bused	14		Reserved, not bused
	15		Reserved, not bused	16		Reserved, not bused
	17		Reserved, not bused	18		Reserved, not bused
	19		Reserved, not bused	20		Reserved, not bused
	21		Reserved, not bused	22		Reserved, not bused
	23		Reserved, not bused	24		Reserved, not bused
	25		Reserved, not bused	26		Reserved, not bused
	27		Reserved, not bused	28		Reserved, not bused
	29		Reserved, not bused	30		Reserved, not bused
	31		Reserved, not bused	32		Reserved, not bused
	33		Reserved, not bused	34		Reserved, not bused
	35		Reserved, not bused	36		Reserved, not bused
	37		Reserved, not bused	38		Reserved, not bused
	39		Reserved, not bused	40		Reserved, not bused
	41		Reserved, bused	42		Reserved, bused
	43		Reserved, bused	44		Reserved, bused
	45		Reserved, bused	46		Reserved, bused
	47		Reserved, bused	48		Reserved, bused
	49		Reserved, bused	50		Reserved, bused
	51		Reserved, bused	52		Reserved, bused
	53		Reserved, bused	54		Reserved, bused
Address	55	A22* (AD16*)	Address bus	56	A23* (AD17*)	Address bus
	57	A20* (AD14*)	[See NOTE (4)]	58	A21* (AD15*)	[See NOTE (4)]
	59		Reserved, bused	60		Reserved, bused

NOTES: (1) All reserved pins are reserved for future use and should not be used if upwards compatibility is desired.
(2) Pins 1–40 — are for "SPECIAL USE". Special uses are defined in categories. Only category-No 1 is currently described: Category No 1 is unconstrained use. Other categories are expected to include higher performance buses, I/O interfaces. etc.
(3) Pins 41–60 — are intended for future address, data, or other PI-related signals, or both.
(4) Address Bus Mnemonics in parentheses are for historical reference.

Electrically conductive components require L_H to be decreased to 0.40 in (10.16 mm).

An example of a typical backplane and the components necessary to implement it are shown in Fig 31.

This section contains only the mechanical specifications for designing an IEEE Std 796 bus interface. The designer must also take into consideration the electrical specifications in Section 3.

4.1.2 IEEE Std 796 Bus Pin Assignments. Printed circuit boards which are designed to interface to the IEEE Std 796 bus have two connectors which plug into the backplane, P1 (Primary) and P2 (Auxiliary). Table 9 shows the pin/signal assignments for the P1 connector on the printed circuit boards. The P1 connector should be bussed as normal signal lines on the backplane. Table 10 shows the pin/signal assignments for the P2 connector on the printed circuit boards. If a backplane is used then the ***reserved*** and ***bussed*** signals should be bussed as normal signal lines.

4.2 IEEE Std 796 Bus Board Form Factors. Certain IEEE Std 796 bus characteristics should be taken into consideration when designing printed circuit boards that interface to it. The designer will ensure himself of IEEE Std 796 bus compatibility if the specifications discussed in this section are followed.

4.2.1 Connector Naming and Pin Numbering Standards. The P1 and P2 connectors on the printed circuit boards designed for the IEEE Std 796 bus interface should adhere to the following standards (see Fig 31).

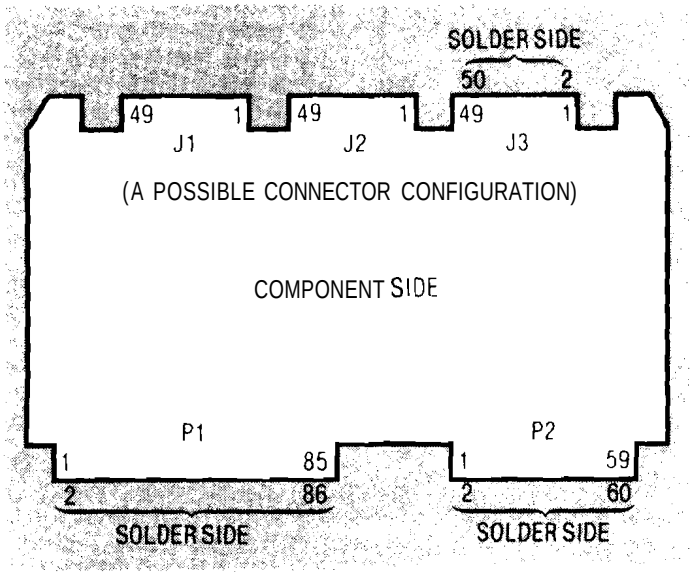
(1) The connectors on the bus side of the board will be called P1 and P2. P1 is the 86 pin main connector and P2 is the 60 pin auxiliary connector.

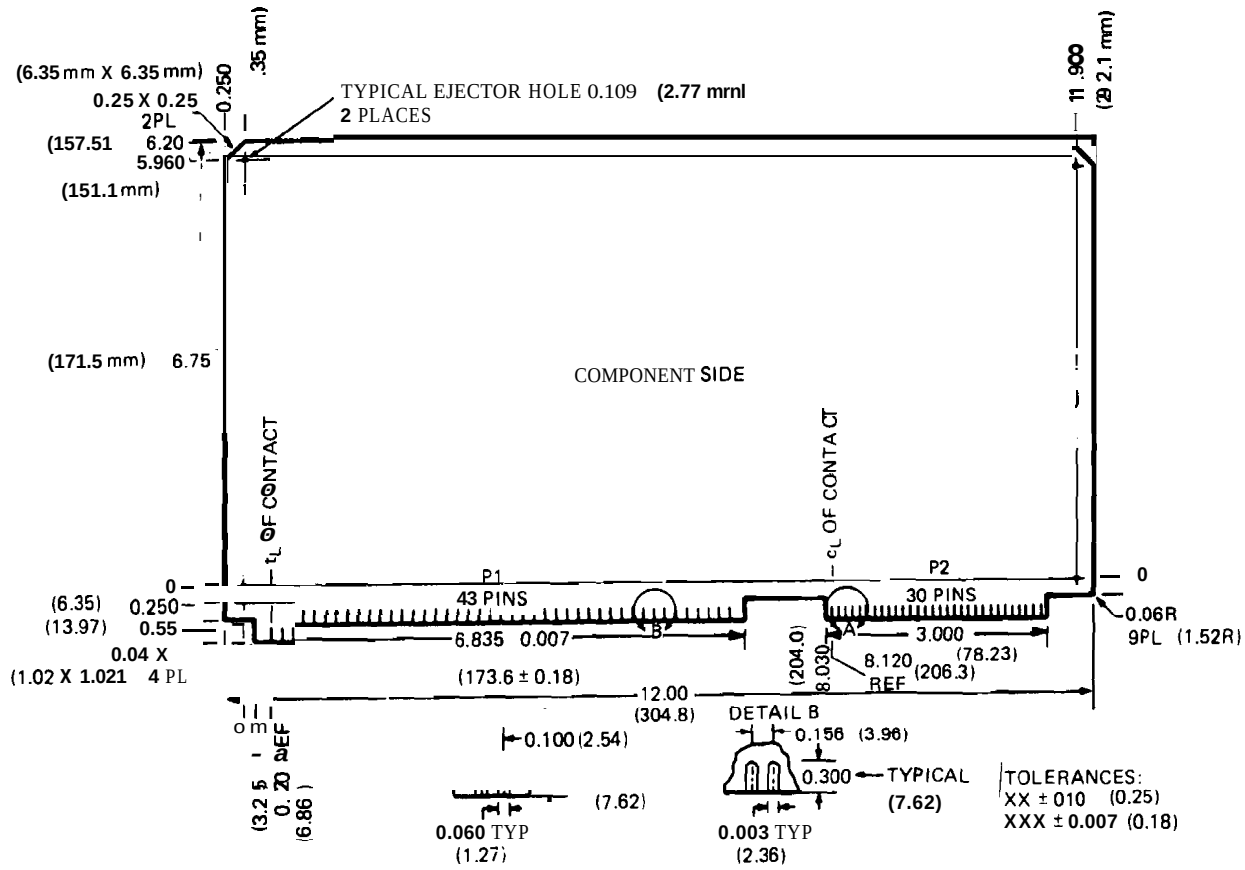
(2) Pins should be numbered with odd number pins on the component side of the board and in ascending order when going counterclockwise around the board (as shown in Fig 32).

4.2.2 Standard Outline of Printed Wiring Boards. Figure 33 shows the standard outline for IEEE Sts 796 bus-compatible boards (printed wire boards and printed circuit boards). The nonbus edge of the board is not restricted. The remainder of the board including connectors P1 and P2 should adhere to the dimensions shown in Fig 33. Only the basic board's standard vertical height is currently specified.

4.2.3 Bus Connectors. The IEEE Std 796 bus backplane has connectors that mate to the P1

Fig 32
Connector and Pin Numbering





NOTE: All dimensions are in inches with millimeters provided in parentheses for reference only. The inch dimensions govern.

Fig 33
Standard Printed Wiring Board Outline

(43/86 pin) board edge connector. The backplane uses 43/86 pin on 0.156 in (3.96 mm) centers connectors.

The P2 connector is a 30/60 pin board edge connector with 0.100 in (2.54 mm) pin centers.

5. Levels of Compliance

This section presents the concept and notation of levels of compliance with the IEEE Std 796 bus standard as follows:

(1) Variable elements of capability composing the essence of IEEE Std 796 bus standard compliance.

(2) General discussion of compliance relationship for masters and for slaves.

(3) Notation for describing level of compliance with the IEEE Std 796 bus standard.

The notion of levels of compliance is introduced to facilitate the use of IEEE Std 796 bus products of varying capability manufactured by diverse vendors. It bounds the variability allowed within the IEEE Std 796 bus specification and provides a succinct and convenient notation for these variables.

5.1 Variable Elements of Capability. The IEEE Std 796 bus is very versatile allowing systems to be constructed with boards of varying capability. The IEEE Std 796 bus allows for variations in data path width, I/O address path

width, and interrupt attributes. In addition it is recognized that some vendors' products have differing memory address path width.

5.1.1 Data Path. The IEEE Std 796 bus allows for 8-bit and 16-bit data path products. The 16-bit data path products use the byte swapping technique described in 2.2.2.4, thus allowing the 8-bit and 16-bit products to work together.

5.1.2 Memory Address Path. The IEEE Std 796 bus standard designates a 14-bit address path. In many systems a 16-bit or 20-bit address path may be sufficient.

5.1.3 I/O Address Path. The IEEE Std 796 bus allows for 8-bit and 16-bit I/O address paths. The 16-bit path products must also be configurable to act as 8-bit path products.

5.1.4 Interrupt Attributes. The IEEE Std 796 bus (see 2.3) allows for considerable variety in interrupt attributes. A product may support no interrupts, non-Bus vectored (NBV) interrupts, two-cycle bus vectored interrupts, and three-cycle bus vectored interrupts. There are two methods of interrupt sensing: the preferred level-triggered, and for historical compatibility only, edge-level-triggered.

(1) ***Level-Triggered.*** The active level of the request line indicates an active request. Requiring no edge to trigger an interrupt allows several sources to be attached to a single request line. Sources for level triggered sense inputs should provide a programmatic means to clear the interrupt request.

(2) ***Edge-Level Triggered.*** The transition from the inactive to the active level indicates an active request if and only if the active level is maintained at least until it has been recognized by the master. The requirement for a transition precludes multiple sources on a request line but edge-level triggering removes the requirement that the source have a programmatic means to clear the interrupt request.

NOTE: Edge-level triggered is described only to allow for historical compatibility. New designs shall use level-triggered interrupt sensing.

A master may support either or both of the above interrupt sensing methods. It is necessary to configure the system so that the sources of the interrupt requests correspond to the interrupt sensing method of the master. Note that a source which is compatible with level triggering is also compatible with edge-level triggering.

5.2 Masters and Slaves. When constructing IEEE 796 bus systems it is not necessary that all modules have identical capabilities. One may for instance have a master with an 8116-bit data path and a slave with an 8-bit data path. The system is completely functional, though the application should restrict itself to 8-bit access to the slave.

The key concept when constructing an IEEE Std 796 bus system is that of required capability versus supplied capability. Each product will provide some set of capability. A transaction between two such products will be restricted to the sets of capability of the two products. In some cases the intersection may be null implying fundamental incompatibility. It is the responsibility of the system designer to ensure the viability of this intersection.

5.3 Compliance Level Notation. A notation is introduced which allows a vendor to succinctly and accurately specify a product's level of compliance with the IEEE Std 796 bus standard. For boards which may act as either masters or slaves, the compliance levels should be specified for both cases. Increasing levels of compliance subsume lesser levels for data path width, memory address path width, and I/O address path width. Interrupt attributes are listed separately as they are independent of one another. The lack of an element (that is, no I/O address path) specification normally implies no capability for this element.

5.3.1 Data Path

D8 represents an 8-bit data path

D16 represents an 8116-bit data path

5.3.2 Memory Address Path

M16 represents a 16-bit memory address path

M20 represents a 20-bit memory address path

M24 represents a 24-bit memory address path

5.3.3 I/O Address Path

I8 represents an 8-bit I/O address path

I16 represents an 8-bit or 16-bit I/O address path

5.3.4 Interrupt Attributes

VO represents non-Bus vectored interrupt requests

V2 represents two-cycle bus vectored interrupt requests

V3 represents three-cycle bus vectored interrupt requests

E represents edge-level triggering only
L represents level triggering
EL represents level or edge-level triggering

The interrupt attributes notation can be concatenated to represent multiple capabilities.

5.3.5 An Example. A versatile combination I/O and memory slave board which supports an 8/16-bit data path, a 20-bit memory address, an 8-bit or 16-bit I/O address, NBV interrupt requests, two- and three-cycle bus vectored interrupt requests is specified as follows:

IEEE Std 796 bus compliance: Slave D16 M20I16 V023 L

5.3.6 Compliance Marking. The compliance levels of a card shall be clearly marked on the printed circuit board and also in the printed specifications.

6. Bibliography

Intel Multibus™ Specification, *Intel Corporation,* Manual Order No 9800683, 1978.